CICS Transaction Server for z/OS

How to configure a z/OS LDAP Server for CICS Development purposes

Robert Harris,
CICS Technical Strategy,
IBM Hursley.

Issued:	01 August 2002
Revision Date:	01 August 2002
Previous Revision Date:	None
Review Date:	As Required

Take Note!

Before using this document be sure to read the general information under "Notices".

First Edition, August 2002.

Notices:

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Summary of amendments

Date Of Change	Change made to document
01/08/2002	Creation

Reference Material and Bibliography:

This document uses a short reference to the following documentation:

Short reference	Book Title
CICS RDO Book	CICS Resource Definition Guide SC34-5722
CICS SDG	CICS System Definition Guide SC34-5725
CICS CG	CICS Customization Guide SC34-5706
CICS EXT	CICS External Interfaces Guide SC33-1944
CICS JAVA	Java™ Applications in CICS SC34-6000-0
CICS APR	CICS Application Programming Reference SC34-5702
CICS SPR	CICS System Programming Reference SC34-5726
JVM Book	New IBM Technology featuring Persistent Reusable Java Virtual Machines SC34-5881
LDAP Admin	z/OS Security Server LDAP Server Administration and Use SC24-5923-02
LDAP Util	z/OS Security Server LDAP Client Programming SC24-5924-01
LDAP Red Book	Understanding LDAP SG24-4986

Preface:

This document is aimed at CICS System Programmers who want to configure a z/OS Lightweight Directory Access Protocol Server for use by CICS Transaction Server for z/OS Version 2.2.

Java™ programmers who are going to implement Enterprise Java Bean™ function in the CICS Transaction Server for z/OS environment need to know about LDAP configuration. Knowledge of Enterprise Bean™ function is, however, not required to get the best out of this document.

It is aimed at taking a System Programmer who is knowledgable about CICS Java environment through the steps needed to configure an host LDAP Server for CICS' usage. Examples are given showing what needs to be done and how to achieve it. An appendix shows how to configure a LDAP server for access to DB2 via JDBC™ 2.0 by CICS.

You do not need any detailed knowledge of CICS to get the best out of this document; however, an appreciation of the mainframe environment is desirable and one needs an appreciation of LDAP and the way it is used by an Enterprise Bean™ in the EJB™ environment provided by CICS.

The information and code in this document is **only** applicable to CICS Transaction Server for z/OS Version 2.2. It is not applicable to earlier CICS releases.

This document uses Colour to highlight items of interest, so access to the PDF as well as the hard copy in the absence of a colour print is desirable.

Table of Contents

List of Figures

LDAP on z/OS for CICS TS Version 2.2

Introduction

This document describes the implementation of a z/OS Lightweight DIrectory Access Protocol Server for use by CICS Transaction Server Version 2.2. It assumes that the LDAP Server has been installed but not yet configured. Instructions and guidance are given under the assumption that the arrangement is for the LDAP Server to be used within a Development environment (as opposed to a Production setup).

A Lightweight Directory Access Protocol Server primarily acts as a dictionary for Enterprise Bean related information, but is actually a general-purpose depository for any type of looked-up information.

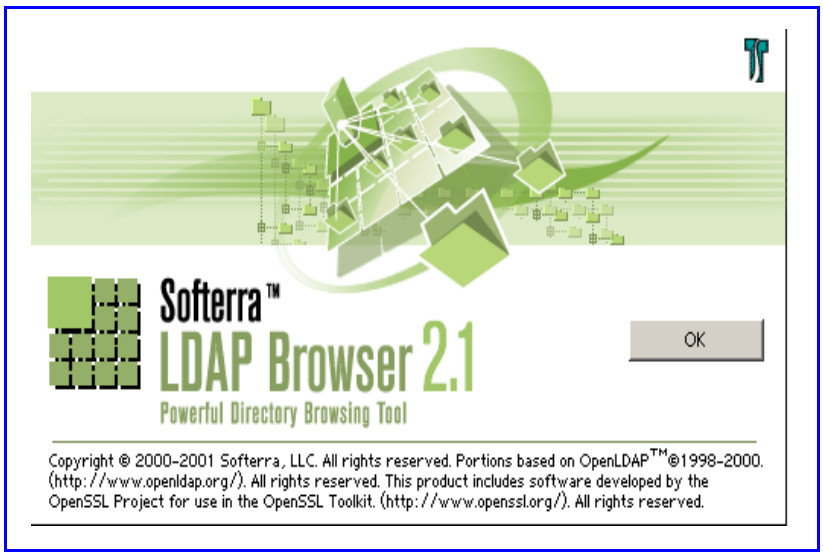
Documentation

- SC24-5923-02: *z/OS Security Server LDAP Server Administration and Use* contains information about configuring a LDAP Server
- SC24-5924-01: *z/OS Security Server LDAP Client Programming* is more of a LDAP programming guide, but it contains documentation for the LDAP utility commands
- SG24-4986: *Understanding LDAP* is a Red Book that describes the LDAP environment and explains concepts

Requirements

The LDAP Server used in this document is the z/OS Version 1 Release 2 Security Server LDAP Server.

You will need a LDAP Browser. The one I use is Softerra LDAP Browser obtainable from `www.shareware.com`.



I also use the IBM Secureway Directory Tool:

Conventions

Throughout this document the following terms will frequently occur:

- LDAP Server address
- LDAP server port
- Administrator Userid
- Administrator Password
- Suffix

To show what needs to be done, these will be set to values used on my z/OS system at Hursley in the UK. Example code and commands are presented using my settings. You will have to use your own values to execute the items in this document.

My settings are:

LDAP Server Address	winmvs2c.hursley.ibm.com
LDAP server port	2389
Administrator userid	cn=admin
Administrator Password	secret
Suffix	ou=RAH,o=IBM Hursley,c=UK
HFS Home directory	/u/rharri1

CICS Documentation

The arrangement discussed in this document is that contained in the *CICS Java* book (SC34-6000-0 *Java Applications in CICS*) from the section relating to LDAP configuration.

How LDAP works

LDAP is based on a naming hierarchy which is governed by the X500 naming structure. This means that all entries are in a `Key=Value` format, with the Key part being governed by the hierarchy. In most cases (but not all) both the Key and the Value are not case sensitive. Consequently, it is wise to assume that they are used in mixed-case mode.

The key is called a Distinguished Name (`dn`). A dn can be made up of several components called Relative Distinguished Names (`rdn`).

Distinguished Names are specified in a left to right sequence of Relative Distinguished Names, with the right-most rdn being the top of the tree. Thus, given a dn of `o=RAH,ou=IBM Hursley,c=uk` there are three rdns: `o`, `ou` and `c` and the `c=uk` rdn is the top of the tree.

The most common element of a dn is the Common Name (`cn`).

This has an immediate implication in supplying Userids: the format to use is `cn=<userid>` and not just the name of its own.

Here are some elements of a dn (each of which is a rdn) at the LDAP V3 level:

cn	Common Name	A persons full name
sn	SurName	
c	Country	In CL2 format
l	Locality	A place, town etc.
st	State	A country etc.
street	Street	
o	Organisation	Company
ou	Organisational Unit	Company subdivision
title	Title	Mr/Ms/Sir etc.

The Red Book `SG24-4986:` *Understanding LDAP* provides a full appreciation of LDAP.

LDAP Configuration choices

You are going to end up with a LDAP configuration that is determined by System Definition and User choice.

The system definition partially allows you to choose (but this will usually be fixed):

- The company naming convention

But you do not have any control over:

- The WebSphere naming convention

The user configuration allows you to choose:

- A departmental point for your definitions (like Test, Acceptance)
- A CICS region-specific point

These choices affect both the LDAP Server and the definitions used within CICS.

Decisions about the user configuration apply because I am building a LDAP hierarchy for the Development environment. Other choices will be made for a Production setup.

LDAP Servers can contain both Test and Production information, but the usual access/security rules will usually mean that Test and Production LDAP servers are different.

I hope that using this document to create a Development LDAP environment will lead to a considered choice for the Production setup.

End result

The end result of your choices will be to build a structure within the LDAP Server. Figure 1 shows the result for choices I have made in this document.

Items with blue markings are those fixed by the System Administrator,

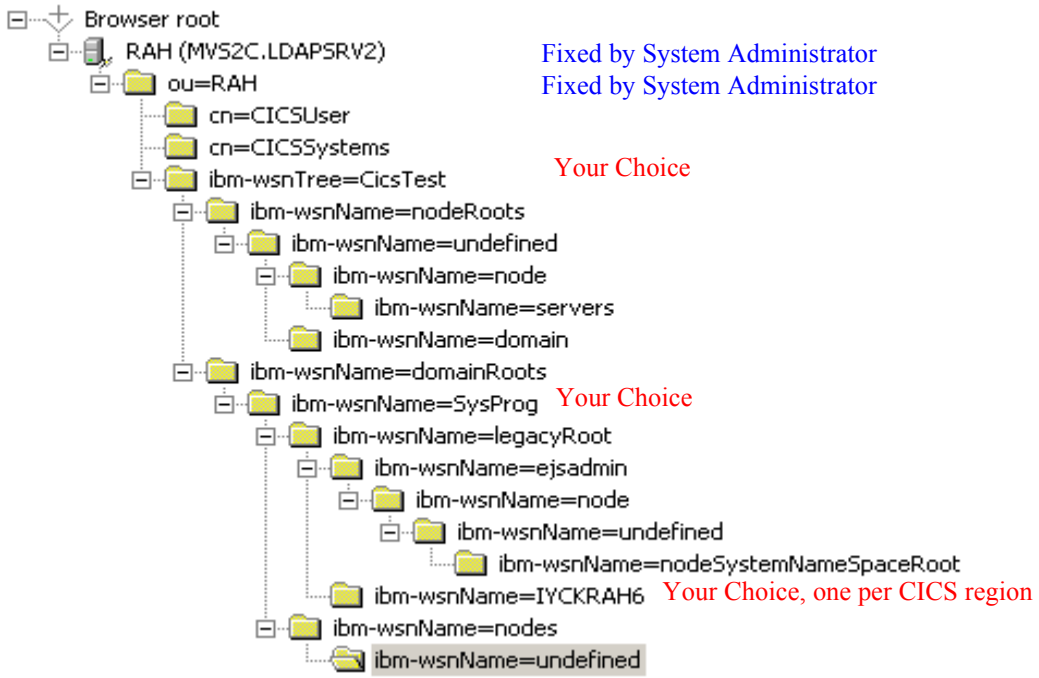


Figure 1: Final LDAP Structure

whilst those with red are freely available.

Items will get added under the `ibm-wsnname=IYCKRAH6` entry.

The Initial Hierarchy

After the LDAP Server Installation, there will be an initial dn naming the company and division for which the LDAP server is going to operate. This is referred to as the suffix. The suffix must be known as it is used for configuration purposes.

This will usually be fixed by the System Administrator, as it will contain company specific details.

JCL

After the LDAP Server has been installed, you will end up with some Started Task JCL which looks like Figure 2 :

```
//*************************************************************
//* Licensed Materials - Property of IBM
//* 5647-A01
//* (C) Copyright IBM Corp. 1997, 1999
//*
//*************************************************************
//*
//* Procedure for starting the LDAPSRV server
//*
//* To start server using configuration file
//* /etc/ldap/slapd.conf specify:
//*  s ldapsrv
//*
//* To start server using alternate configuration file or
//* other parameters specify:
//*  s ldapsrv,parms='options'
//* where options can be:
//*     -f filename  # alternate configuration file
//*     -d level  # debug level (65535 turns on all debugs)
//*     -p portno # non-secure port number
//*     -s portno # secure port number
//*
//* An alternative to the -f option is to define a CONFIG DD.
//* The remaining options are optional.  If not set, message/debug
//* levels are set to 0, non-secure port number will be 389, and
//* secure port number will be 636.  NOTE:  use of these low port
//* numbers will require that the LDAPSRV server run under a userid
//* that has OpenEdition UID 0.
//*-----------------------------------------------------------
//* CONFIG can be used to specify the LDAP server config file.
//* ENVVAR can be used to specify any environment variables
//* DSNAOINI can be used to specify the file required by DB2.
//*-----------------------------------------------------------
//LDAPSRV  PROC PARMS='',REGSIZE=64M
//*-------------------------------------------------------------
//LDAP  EXEC PGM=GLDSLAPD,REGION=&REGSIZE,TIME=1440,
//   PARM=('/&PARMS >DD:SLAPDOUT 2>&1')
//STEPLIB  DD DSN=PP.LDAP.ZOS120.SGLDLNK,DISP=SHR
//   DD DSN=SYS2.DB2.V710.SDSNLOAD,DISP=SHR
//CONFIG   DD PATH='/etc/ldapsrv2/slapd.conf'
//DSNAOINI DD DSN=PP.LDAP.ZOS120.LDAPSRV2.DSNAOINI,DISP=SHR
//SLAPDOUT DD SYSOUT=A
//SYSOUT   DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//CEEDUMP  DD SYSOUT=A
```

Figure 2: LDAP Server JCL

Here are the things to notice:

- `//DSNAOINI` refers to a required DB2 file
- `//CONFIG` statement refers to a HFS file (`/etc/ldapsrv2/sladp.conf`) containing the configuration for the LDAP Server which contains the port number for access
- Information is displayed using Streams which is directed to a `SYSOUT` file (`>DD:SLAPDOUT 2/&1`)

The HFS Configuration file

The main configuration file is held within HFS. It should look something like Figure 3 (which has lots of comments removed):

```
#
# Connection Info
#
port    2389
secureport 3389
security   none
#
# Volume Controls
#
validateincomingV2strings yes
sendV3stringsoverV2as  UTF-8
verifySchema     on
sizeLimit     500
timeLimit     3600
maxConnections   200
maxThreads    200
waitingThreads   10
verifySchema     on
validateincomingV2strings yes
sendV3stringsoverV2as  UTF-8
#
# DB2 Info
#
database tdbm  GLDBTDBM
servername  DSN710RH
dbuserid    LDAPSR2
databasename   LDAPDBRH
#
# Administrator definition
#
adminDN    "cn=admin"
adminPW  secret
#
# Top Level Definition
#
suffix "ou=RAH,o=IBM Hursley,c=UK"
#
# ---------------------------------------
#
# adminDN <distinguishedname>
#
# Example:
#   adminDN "cn=Admin, o=Your Company"
#
#   The adminDN option should be updated to contain a
#   distinguished name within one of the suffixes defined below.
#   This requires that an entry exist in the directory for this
#   distinguished name and it will be used when evaluating an
#   LDAP bind operation for the AdminDN.
#
# ---------------------------------------
# suffix <toplevelname>
#
# Default Value: none
#
# Example:
#   suffix "o=Your Company"
#
```

Figure 3: LDAP initial configuration file

This configuration file is just sufficient to enable the LDAP Server to be started. More actions are taken within the LDAP Server for it to become useful.

Advanced configuration options can be used to control replication and referral (linkage of LDAP Server instances to form a larger entity), but these are outside the scope of this document. I assume that the LDAP Server is going to be used in a Development environment, and that many individual CICS regions are going to use the same LDAP Server instance without interfering with each other.

Apart from the DB2 information (the LDAP Server uses a DB2 database to hold information) the main things to note are:

- The Port number (`port`) used to communicate with the Server
- The name (`adminDN`) and password (`adminPW`) used for communicating with the LDAP Server
- The suffix ("ou=RAH,o=IBM Hursley,c=UK") used to define the LDAP namespace

The `port`, `adminDN` and `adminPW` items, together with the IP Address of the z/OS system are needed to contact the LDAP Server. (See “Nameserver (`com.ibm.cics.ejs.nameserver`)” on page 42.)

The suffix of `"ou=RAH,o=IBM Hursley,c=UK"` has to be specified in quotes and forms the dn of the LDAP namespace being processed. The suffix will usually be set by the System Administrator.

Starting the LDAP Server

When you start the LDAP Server, the following messages should appear in the Job Log:

```
GLD4005I Environment variable file not found. Environment variables not set.  Continuing.
GLD0022I z/OS Version 1 Release 2 Security Server LDAP Server
 Starting slapd.
GLD0010I Reading configuration file //DD:CONFIG.
GLD0053I Configuration read security of none.
GLD0185I Connections allowed only on the nonsecure port.
GLD0163I Backend capability listing follows:
GLD0166I Backend type: tdbm, Backend ID: TDBM BACKEND, Backend suffix:
OU=RAH,O=IBM HURSLEY,C=UK::
GLD0165I Capability: LDAP_Backend_ID  Value: TDBM BACKEND
GLD0165I Capability: LDAP_Backend_BldDateTime  Value: 2001-12-04-14.59.32.000000
GLD0165I Capability: LDAP_Backend_APARLevel  Value: LDAP
GLD0165I Capability: LDAP_Backend_Release  Value: R 2.0
GLD0165I Capability: LDAP_Backend_Version  Value: V 1.0
GLD0165I Capability: LDAP_Backend_Dialect  Value: DIALECT 1.0
GLD0165I Capability: LDAP_Backend_BerDecoding  Value: BINARY
GLD0165I Capability: LDAP_Backend_ExtGroupSearch  Value: YES
GLD0165I Capability: LDAP_Backend_krbIdentityMap  Value: YES
GLD0165I Capability: supportedControl  Value: 2.16.840.1.113730.3.4.2
GLD0165I Capability: supportedControl  Value: 1.3.18.0.2.10.2
GLD0167I End of capability listing for Backend type: tdbm, Backend ID: TDBM BACKEND,
Backend suffix: OU=RAH,O=IBM HURSLEY,C=UK.
GLD0164I Backend capability listing ended.
GLD0002I Configuration file successfully read.
GLD0189I Nonsecure communication is active for IP: INADDR_ANY, nonsecure port: 2389.
GLD0122I Slapd is ready for requests.
```

This means that the LDAP Server will accept requests.

Issuing MVS commands to the LDAP Server

The LDAP Server is a started task, and so will accept MVS Modify commands (`/F <jobname>`) to control its running. See “LDAP Server tracing” on page 62 for details.

Configuring the LDAP Browser and Directory Tool

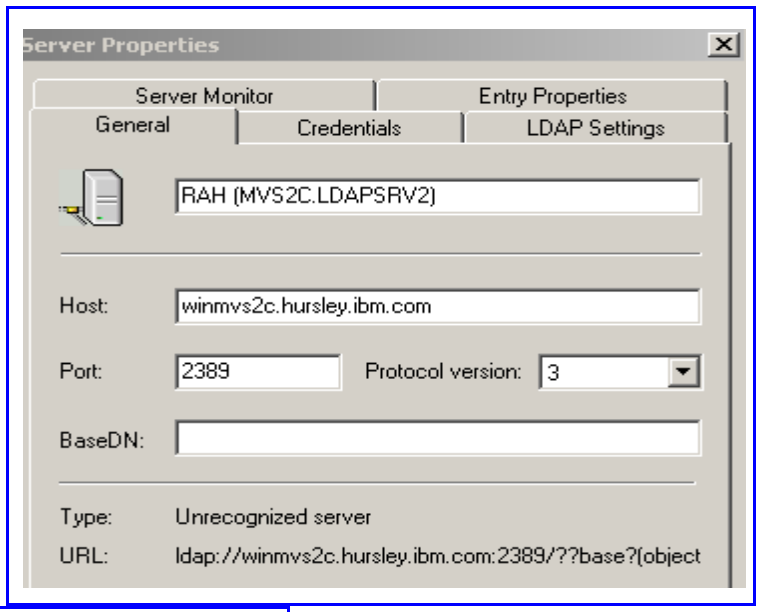
In order to access and configure the LDAP Browser and the Directory Tool you will need:

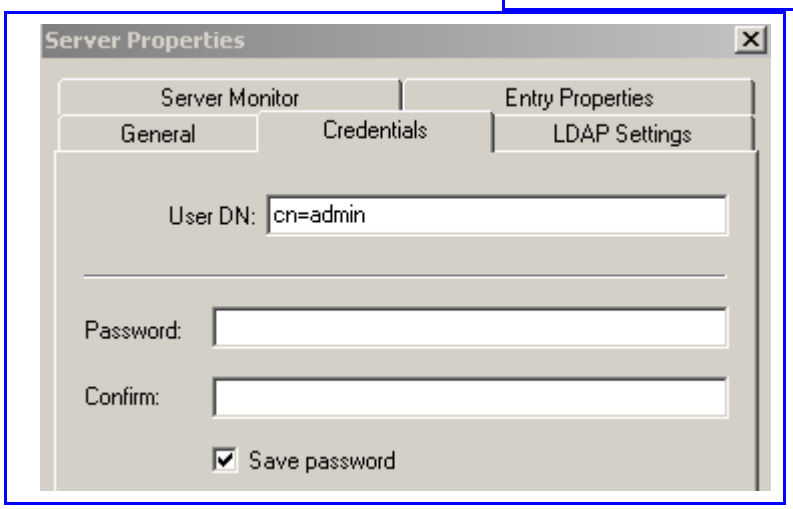
- The IP Address of the z/OS hosting the LDAP Server
- The Port number for access - from the `port` setting
- The Userid and Password for administration purposes - from the `adminDN` and `adminPW` settings

It's important that the full dn format `(cn=admin)` is used for the Userid!

These values feed into `ldapmodify` and `ldapadd` commands that define items in the LDAP server. These commands are issued from with the z/OS Unix System Services shell and it is usually convenient to create shell scripts to issue these commands. `SC24-5924-01:` *`z/OS Security Sever LDAP Client Programming`* contains information about these commands.

After properties configuration the settings of my LDAP Browser are:





And the IBM Secureway Directory Management tool settings are:

Connect to directory server

Server name : **ldap://**	winmvs2c.hursley.ibm.com
Port :	2389
User DN :	cn=admin
User password :	******
☐ Use SSL	
Keyclass file name :	
Keyclass file password :	

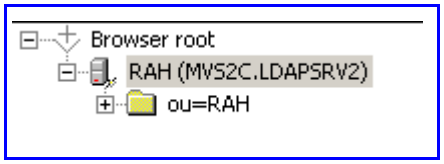
The configuration file for the Directory Management tool is:

```
#browser=
server1.url=ldap://winmvs2c.hursley.ibm.com:2389
server1.security.bindDN=cn=admin
server1.security.password=secret
#server1.security.ssl.keyclass=
#server1.security.ssl.keyclass.password=
```

Contacting the LDAP Server

Once the LDAP Browser has been configured (and the LDAP Server contacted), it should display a very simple structure.

The left hand side will show the initial structure:

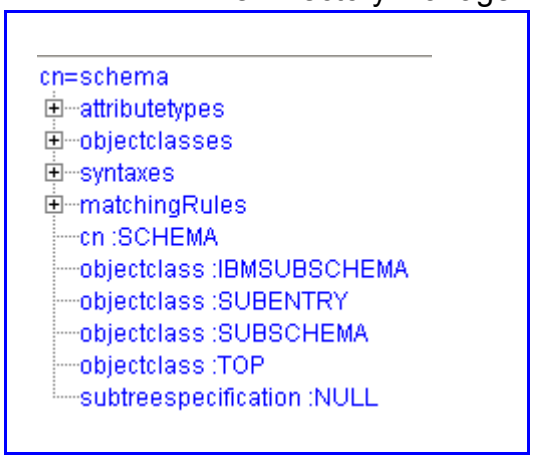


whilst the right hand side will show the contents:

Name	Value	Type	Size
ou	RAH	entry	unknown
ibmdirectoryversion	z/OS V1R2	text attribute	9
supportedcontrol	2.16.840.1.113730.3.4.2	operational attribute	23
supportedcontrol	1.3.18.0.2.10.2	operational attribute	15
namingcontexts	ou=RAH,o=IBM Hursley,C=UK	operational attribute	25
subschemasubentry	CN=SCHEMA,ou=RAH,o=IBM Hursley,C=UK	operational attribute	35
supportedsaslmechanisms	EXTERNAL	operational attribute	8
supportedldapversion	2	operational attribute	1
supportedldapversion	3	operational attribute	1

You can see that the suffix has appeared in the `subschemasubentry` item and a folder has appeared for the `ou` being used.

The Directory Management tool shows:



The LDAP Server now needs to have a basic structure (called a schema) added. Continue at “Install the Schema” on page 14.

The WebSphere naming schema

The LDAP Server needs to have a schema. A schema defines the structure of the database and several structures are possible. The preferred schema is that used by the IBM WebSphere product - even if you do not intend to use WebSphere itself. This is the structure that I am going to use for my LDAP Server.

This WebSphere schema definition is available in `/usr/lpp/ldap/etc/WebSphereNaming.ldif`. Alternatively, it is shipped with CICS TS 2.2 in `/usr/lpp/cicsts/cicsts22/utils/namespace/WebSphereNamingSchema.ldif` (however, this latter is affected by APARs, so use the WebSphere supplied version if available).

You should copy this file and rename to `MyWebSphereNamingSchema.ldif`.

It should look like Figure 4 (initial part only).

```
# ---------------------------------------------------------------
# This file is shipped in code page IBM-1047 and must remain in
# code page IBM-1047.
# ---------------------------------------------------------------

dn:cn=schema, <suffix>
changetype: modify
add: attributetypes
attributetypes: (
  1.3.18.0.2.4.1102
  NAME 'ibm-wsnEntryType'
  DESC 'Defines the type of WebSphere Name Tree entry'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
  SINGLE-VALUE
  USAGE userApplications
  )
ibmattributetypes: (
  1.3.18.0.2.4.1102
  DBNAME( 'ibmwsnEntryType'  'ibmwsnEntryType' )
  ACCESS-CLASS normal
  LENGTH 32
  EQUALITY
  )
attributetypes: (
  1.3.18.0.2.4.1103
  NAME 'ibm-wsnName'
  DESC 'Name of an entry in the WebSphere Name Tree'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
  USAGE userApplications
  )
ibmattributetypes: (
  1.3.18.0.2.4.1103
  DBNAME( 'ibmwsnName'  'ibmwsnName' )
  ACCESS-CLASS normal
  LENGTH 240
  EQUALITY
  )
```

Figure 4: WebSphere naming schema

A `ldif` file contains commands for the `ldapmodify` or `ldapadd` utility.

You have to change the `dn:cn=schema,<suffix>` line by inserting the dn of the suffix. In my case, it will look like

```
dn:cn=schema, ou=RAH, o=IBM Hursley, c=UK
```

Note that quotes are not required inside a ldif file whereas they are in executing a ldapmodify command.

Installing the WebSphere schema

The WebSphere schema is installed by operations within the OpenEdition Shell. Again you will need:

- The IP Address of the z/OS hosting the LDAP Server
- The Port number for access - from the `port` setting
- The Userid and Password for administration purposes - from the `adminDN` and `adminPW` settings

Whilst the LDAP server is running, within the OpenEdition shell issue the following command on the modified schema file:

```
ldapmodify
            -h <hostname>
            -p <portnumber>
            -D <userid>
            -w <password>
            -f /u/rharri1/MyWebSphereNamingSchema.ldif
```

It's important that the full `cn=admin` (or whatever is specified in the LDAP Server configuration file) is used for the Userid and that the `-D` parameter is supplied within quotes (`-D "cn=admin"`). If the command is spread over several lines, you will need to add the \ continuation character at the end of all lines apart from the last one.

If you get a message implying that type or values already exist, then someone else has already done this step for you.

You can see what has been installed by running (in the OE shell):

```
ldapsearch
            -h <hostname>
            -p <portnumber>
            -D <userid>
            -w <password>
            -b "cn=schema,<suffix>"
               "objectclass=*"
```

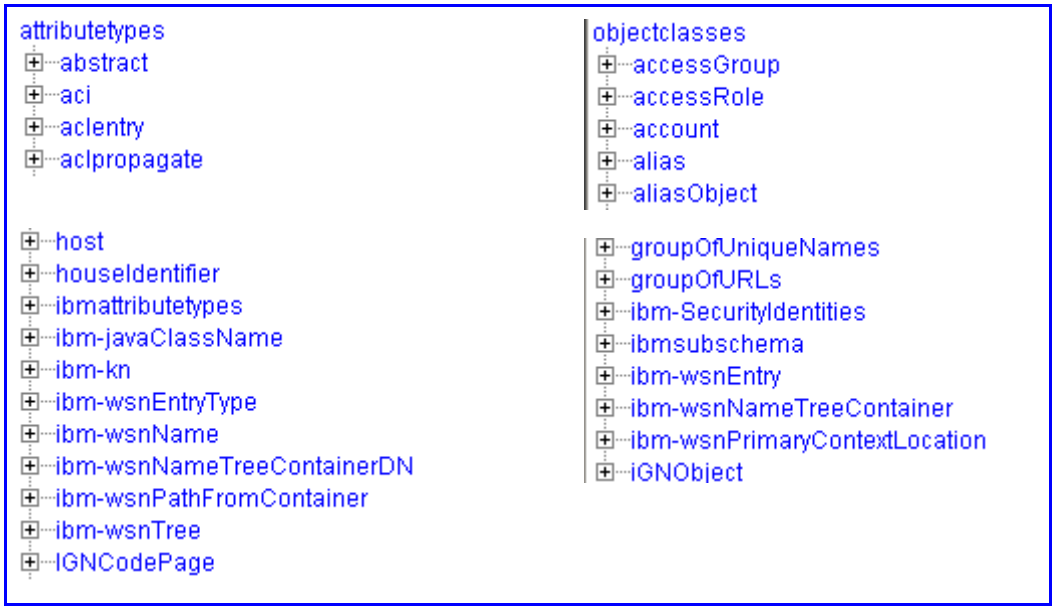
(So it's `-b "cn=schema,ou=RAH,o=IBM Hursley,c=UK"` in my case.)

You can check that everything has been created by placing the following code into a script file (remember to `chmod a+rwx` it and, maybe, changing £s to $s and ¬s to ^s etc. together with the apt namings) and seeing that things match.

```
#
# Shell script to verify WebSphereNamingSchema
#

pserver="winmvs2c.hursley.ibm.com"
pport="2389"
puserid="cn=admin"
ppassword="secret"
pschema="ou=RAH, o=IBM Hursley, c=UK"

echo
echo " --WebSphereNamingSchema Input--"

cat MyWebSphereNamingSchema.ldif    | \
    awk '/NAME .ibm/     {print £0}   \
         /DBNAME\( .ibm/ {        } '


echo " --Attributes--"

ldapsearch -h £pserver  -p £pport      \
           -D £puserid  -w £ppassword  \
           -b "cn=schema,£pschema"     \
              "objectclass=*"        | \
    awk '/¬attr/     {print £0}  '   | \
    awk '/ibm/       {print £0}  '   | \
    awk '/NAME..ibm/ {print £0}  '   | \
    awk 'BEGIN { FS = " "} ;                          \
         { for (i=1;i<=NF;i++)                        \
               { j = i+1 ;                            \
                 m = match(£i,/NAME/) ;               \
                 if ( m !=0 )                         \
                      { print " ", £i, £j ; break }   \
               }                                      \
         }                                         '|\
    cat


echo " --Objects--"

ldapsearch -h £pserver  -p £pport      \
           -D £puserid  -w £ppassword  \
           -b "cn=schema,£pschema"     \
              "objectclass=*"        | \
    awk '/¬object/   {print £0}  '   | \
    awk '/ibm/       {print £0}  '   | \
    awk '/NAME..ibm/ {print £0}  '   | \
    awk 'BEGIN { FS = " "} ;                          \
         { for (i=1;i<=NF;i++)                        \
               { j = i+1 ;                            \
                 m = match(£i,/NAME/) ;               \
                 if ( m !=0 )                         \
                      { print " ", £i, £j ; break }   \
               }                                      \
         }                                         '|\
    cat

#
# End of Shell script
```

Figure 5: Shell script for checking WebSphere schema

Alternatively, check out the schema definitions with the Directory tool:



At this point you have inserted definitions into the LDAP Database, but nothing is actually using them.

Next you have to add the suffix definition into the LDAP structure. Continue at “Creating the Suffix” on page 18.

Why you need to do this

The previous operations have merely configured the LDAP Server without actually placing anything useful within. You have to add an initial entry corresponding to the `suffix` so that everything else can use this as the base for further definitions.

Creating the entry

Create a ldif file for the addition of the suffix (I've called it `Mysuffix.ldif`). It should contain the left-most rdn of the `suffix` entry (which is `ou` in my case):

```
dn: ou=RAH,o=IBM Hursley,c=uk
objectclass: organizationalunit
ou: RAH
```

Figure 6: Creating the Suffix

Observe that it is the `ou` part of the suffix (the left-most) that is the required entry but the whole of the suffix is quoted in the dn field. The suffix is inserted by doing a:

```
ldapadd
        -h <hostname>
        -p <portnumber>
        -D <userid>
        -w <password>
        -f /u/rharri1/Mysuffix.ldif
```

Checking it made it

In the LDAP Browser (after rebinding) the OU folder now contains the entry:

Name	Value	Type	Size
objectclass	organizationalunit	text attribute	18
objectclass	TOP	text attribute	3
ou	RAH	text attribute	3
createtimestamp	20020305120740.377369Z	operational attribute	22
modifytimestamp	20020305120740.377369Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

If you use the Directory Tool (after refreshing), and hit the ACL Button, you can see the permissions associated with the entry:

Subject			Granted rights						
Remove	Distinguished name	Type	Add	Delete	Class	Read	Write	Search	Compare
☐	CN=ADMIN	access-id	☐	☐	Normal	☑	☑	☑	☑
					Sensitive	☑	☑	☑	☑
					Critical	☑	☑	☑	☑
☐	CN=ANYBODY	group	☐	☐	Normal	☑	☐	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐
☐	CN=AUTHENTICATED	group	☐	☐	Normal	☑	☐	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐

You can see that everybody has read access to LDAP Information but only the administrator can manipulate items. Next you create some Userids for CICS usage as shown in Section "Adding the CICS Users" on page 23.

If, however, the ACL display looks like this:

Source DN: ou=RAH,o=IBM Hursley,c=uk

☐ Remove ACL and inherit from ACL source

Reset

☑ Allow descendant directory entries to inherit ACL from this entry

Subject			Granted rights						
Remove	Distinguished name	Type	Add	Delete	Class	Read	Write	Search	Compare
☐	NORMAL:RSC:SYSTEM:RSC	cn=anybody	☐	☐	Normal	☐	☐	☐	☐
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐

then the default acl group has not been correctly setup, and you must fix this as described in Section "Correcting the default" on page 21.

Checking that the correct default permissions have been created

Why you have to do this

Most LDAP Servers will have already created the default access control list (acl) for the system at installation time. However, it is important that this default setting has been setup as a Group (as opposed to an Userid).

Checking the default is a group

You should issue the following command (from within the OE shell):

```
ldapsearch                                  \
           -h <hostname>                    \
           -p <portnumber>                  \
           -D <userid>                      \
           -w <password>                    \
           -b "ou=RAH,o=IBM Hursley,c=UK"   \
            "(objectclass=*)"               \
            aclentry aclpropagate aclsource \
            entryowner ownerpropagate       \
            ownersource
```

(with your own suffix in `-b`).

If it produces something like:

```
entryOwner ownerpropagate ownersource
ou=RAH,o=IBM Hursley,c=uk
aclentry=cn=anybody:NORMAL:RSC:SYSTEM:RSC
ownerpropagate=TRUE
entryowner=access-id:CN=ADMIN
aclsource=ou=RAH,o=IBM Hursley,c=uk
ownersource=default
```

you have got a problem with the default access and must correct it.

The crucial indication of the error is the red `aclentry=cn=anybody:NORMAL:RSC:SYSTEM:RSC` line which shows that the entry is for a specific user and not a group.

However if you get something like:

```
ou=RAH,o=IBM Hursley,c=uk
aclentry=access-id:CN=ADMIN:normal:rwsc:
         sensitive:rwsc:critical:rwsc:
         restricted:rwsc:system:rwsc
aclentry=group:CN=ANYBODY:normal:rsc:system:rsc
aclentry=group:CN=AUTHENTICATED:normal:rsc:system:rsc
ownerpropagate=TRUE
entryowner=access-id:CN=ADMIN
aclsource=default
ownersource=default
```

things are correctly setup, and you need not take any more action in this section. Next you need to add some Userids for CICS access to the LDAP Server: goto "Adding the CICS Users" on page 23.

Correcting the default

You must get rid of the userid entry for `cn=anybody` which will allow the group to become active. At the OE prompt issue a:

```
ldapcp                                          \
           -h <hostname>                        \
           -p <portnumber>                      \
           -d <userid>                          \
           -w <password>                        \
           "acl delete \"ou=RAH,o=IBM Hursley,c=UK\" "
```

with the correct suffix (note the escaped double quotes and the lower-case `-d`).

See what has happened by reissuing the display command):

```
ldapsearch                                      \
           -h <hostname>                        \
           -p <portnumber>                      \
           -D <userid>                          \
           -w <password>                        \
           -b "ou=RAH,o=IBM Hursley,c=UK"       \
            "(objectclass=*)"                   \
            aclentry aclpropagate aclsource     \
            entryowner ownerpropagate           \
            ownersource
```

If it produces something like:

```
ou=RAH,o=IBM Hursley,c=uk
aclentry=access-id:CN=ADMIN:normal:rwsc:
         sensitive:rwsc:critical:rwsc:
         restricted:rwsc:system:rwsc
aclentry=group:CN=ANYBODY:normal:rsc:system:rsc
aclentry=group:CN=AUTHENTICATED:normal:rsc:system:rsc
ownerpropagate=TRUE
entryowner=access-id:CN=ADMIN
aclsource=default
ownersource=default
```

Then the problem has been corrected. The green lines show that the default access groups have been correctly defined.

Once this default acl as a group is around, you create some userids for CICS usage as described in “Adding the CICS Users” on page 23.

Why these are needed

CICS requires two LDAP-sourced identities. One is for CICS system use (CICSUser) and the other (CICSSystems) for general access to the LDAP server.

Creating the Users

There are some definitions in the `/usr/lpp/cicsts/cicsts22/utils/namespace/dfhsns.ldif` file. Copy this file to `Mydfhsns1.ldif`, insert the suffix and remove other definitions so it looks like:

```
# Add the CICSUser (admin) user with the default password
dn: cn=CICSUser, ou=RAH, o=IBM Hursley, c=UK
changetype: add
objectclass: person
cn: CICSUser
sn: CICS Transaction Server 2.2 admin
userPassword: secret

# Add the CICSSystems (runtime) user with the default password
dn: cn=CICSSystems, ou=RAH, o=IBM Hursley, c=UK
changetype: add
objectclass: person
cn: CICSSystems
sn: CICS Transaction Server 2.2 runtime
userPassword: secret
```

Figure 7: Adding CICS userids

The CICSUser entry is used by CICS to access the LDAP Server and so the Userid (see “LDAP access Userid (java.naming.security.principal)” on page 43) and Password (see “LDAP access Password (java.naming.security.credentials)” on page 43) are specified to CICS.

Run this file through `ldapmodify` in the usual fashion:

```
ldamodify  -v                                        \
           -h winmvs2c.hursley.ibm.com -p 2389       \
           -D "cn=admin"               -w secret     \
           -f /u/rharri1/Mydfhsns1.ldif
```

The LDAP Browser (once you have rebound) will now show the new entries:

Name	Value	Type	Size
cn	CICSUser	entry	unknown
cn	CICSSystems	entry	unknown
objectclass	organizationalunit	text attribute	18
objectclass	TOP	text attribute	3
ou	RAH	text attribute	3
createtimestamp	20020328121835.825897Z	operational attribute	22
modifytimestamp	20020328121836.588823Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

and for each created userid

Name	Value	Type	Size
objectclass	person	text attribute	6
objectclass	TOP	text attribute	3
cn	CICSUser	text attribute	8
sn	CICS Transaction Server 2.2 admin	text attribute	33
userpassword	secret	password	6
createtimestamp	20020328121836.750273Z	operational attribute	22
modifytimestamp	20020328121836.750273Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Name	Value	Type	Size
objectclass	person	text attribute	6
objectclass	TOP	text attribute	3
cn	CICSSystems	text attribute	11
sn	CICS Transaction Server 2.2 runtime	text attribute	35
userpassword	secret	password	6
createtimestamp	20020328121836.806348Z	operational attribute	22
modifytimestamp	20020328121836.806348Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Observe that the authorities do not show up on the Browser panel.

If you do a:

```
ldapsearch                                  \
           -h <hostname>                    \
           -p <portnumber>                  \
           -D <userid>                      \
           -w <password>                    \
           -b "ou=RAH,o=IBM Hursley,c=UK"   \
            "(objectclass=*)"               \
            aclentry aclpropagate aclsource \
            entryowner ownerpropagate       \
            ownersource
```

(with the relevant schema) you should see both entries have authorities inherited from the default groups in addition to those especially set (I've split a few lines for readability):

```
ou=RAH,o=IBM Hursley,c=uk
aclentry=access-id:CN=ADMIN:normal:rwsc:sensitive:rwsc:
                   critical:rwsc:restricted:rwsc:system:rwsc
aclentry=group:CN=ANYBODY:normal:rsc:system:rsc
aclentry=group:CN=AUTHENTICATED:normal:rsc:system:rsc
ownerpropagate=TRUE
entryowner=access-id:CN=ADMIN
aclsource=default
ownersource=default

cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK
aclentry=access-id:CN=ADMIN:normal:rwsc:sensitive:rwsc:
                   critical:rwsc:restricted:rwsc:system:rwsc
aclentry=group:CN=ANYBODY:normal:rsc:system:rsc
aclentry=group:CN=AUTHENTICATED:normal:rsc:system:rsc
ownerpropagate=TRUE
entryowner=access-id:CN=ADMIN
aclsource=default
ownersource=default

cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK
aclentry=access-id:CN=ADMIN:normal:rwsc:sensitive:rwsc:
                   critical:rwsc:restricted:
rwsc:system:rwsc
aclentry=group:CN=ANYBODY:normal:rsc:system:rsc
aclentry=group:CN=AUTHENTICATED:normal:rsc:system:rsc
ownerpropagate=TRUE
entryowner=access-id:CN=ADMIN
aclsource=default
ownersource=default
```

You can see authorities using the Directory Tool:

Subject			Granted rights						
Remove	Distinguished name	Type	Add	Delete	Class	Read	Write	Search	Compare
☐	CN=ADMIN	access-id	☐	☐	Normal	☑	☑	☑	☑
					Sensitive	☑	☑	☑	☑
					Critical	☑	☑	☑	☑
☐	CN=ANYBODY	group	☐	☐	Normal	☑	☐	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐
☐	CN=AUTHENTICATED	group	☐	☐	Normal	☑	☐	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐

Next you need to create the required WebSphere tree structure for access as described in “Creating the WebSphere tree structure” on page 27.

Why you need to do this

You need to create the LDAP tree structure under which all definitions are held. It's called a WebSphere tree as it uses the definitions supplied earlier in the WebSphere schema. However, WebSphere itself is not around.

Defining the WebSphere tree structure initially involves creating an anchor point under the previously defined base point (the suffix).

You choose this anchor point name. It should relate to something like Test or Acceptance. I am going to call my anchor point CicsTest.

See "Container Distinguished Name (`com.ibm.ws.naming.ldap.containerdn`)" on page 42 for how this is specified to CICS.

Creating the Tree anchor

There are some definitions in the `/usr/lpp/cicsts/cicsts22/utils/namespace/dfhsns.ldif` file. Copy this file to `Mydfhsns2.ldif`, insert the suffix and remove other definitions so it looks like:

```
# Build the name tree container
# This matches the defaults supplied by Websphere for zOS
dn: ibm-wsnTree=CicsTest, ou=RAH, o=IBM Hursley, c=UK
changetype: add
objectclass: ibm-wsnNameTreeContainer
ibm-wsnTree: CicsTest
```

Figure 8: Creating the WebSphere Tree anchor

As the tree structure I am going to use is `CicsTest` it is this name that is used in the `dn` clause and the associated `ibm-wsnTree` entry.

The `dn` of `ibm-wsnTree=CicsTest, ou=RAH, o=IBM Hursley, c=UK` is referred to as the containerdn.

Run this file through `ldapmodify` in the usual fashion:

```
ldamodify  -v                                    \
           -h winmvs2c.hursley.ibm.com -p 2389   \
           -D "cn=admin"               -w secret \
           -f /u/rharri1/Mydfhsns2.ldif
```

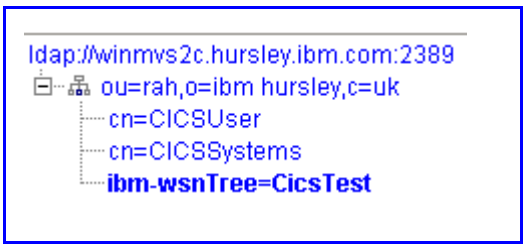
The LDAP Browser (once you have rebound) will now show the new entries:

Name	Value	Type	Size
cn	CICSUser	entry	unknown
cn	CICSSystems	entry	unknown
ibm-wsnTree	CicsTest	entry	unknown
objectclass	organizationalunit	text attribute	18
objectclass	TOP	text attribute	3
ou	RAH	text attribute	3
createtimestamp	20020328123306.800069Z	operational attribute	22
modifytimestamp	20020328123307.325656Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

and for the ibm-wsnTree entry of CicsTest:

Name	Value	Type	Size
objectclass	ibm-wsnNameTreeContainer	text attribute	24
objectclass	TOP	text attribute	3
ibm-wsntree	CicsTest	text attribute	8
createtimestamp	20020328123307.762942Z	operational attribute	22
modifytimestamp	20020328123307.762942Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

The Directory tool also shows the new entry:



Once this anchor is in place, the required tree structure can be created under it. Continue with “Creating the Tree Structure” on page 29.

Creating the Tree Structure

The name of the Tree Structure is upto you. I am going to call mine SysProg. See "Anchor point (`com.ibm.ws.naming.ldap.noderootrdn`)" on page 43 for how this is specified to CICS.

This structure is created by running a CICS-supplied script which invokes a java class. This script needs amending to indicate HFS directories.

The script itself is in the `/usr/lpp/cicsts/cicsts22/utils/namespace/DFHBuildSNS` file. Copy this as `MyDFHBuildSNS` (ensuring the execute permission is set). It should look like:

```
# Call the Java program that will build the SNS

# If executing this utility from the location where
# it is shipped with CICS, it requires NO changes
# in order to run.
#
# If executing it from another location, alter the
# next environment variable value to point to
# the base HFS directory where CICS is installed.
#
# For example, /usr/lpp/cicsts/cicsts22
export CICS_HOME=../..

################################################################
# Do not change anything below this line
################################################################
export CLIB=£CICS_HOME/lib

# Build the correct classpath
export BUILT_CP=$CICS_HOME/utils/namespace/dfhnsutils.jar
export BUILT_CP=$BUILT_CP:$CLIB/security/dfhreg.jar
export BUILT_CP=$BUILT_CP:$CLIB/dfjname.jar:$CLIB/websphere.jar
export BUILT_CP=$BUILT_CP:$CLIB/dfjcicsras.jar
export BUILT_CP=$BUILT_CP:$CLIB/ras.jar:$CLIB/dfjwrap.jar

java -cp £BUILT_CP com.ibm.cics.naming.utils.DFHBuildSNS -Xmx5M £"@"
```

Change the `export CICS_HOME` line to be the HFS directory used for installing CICS. So the script should look like:

```
export CICS_HOME=/usr/lpp/cicsts/cicsts22

################################################################
# Do not change anything below this line
################################################################
export CLIB=£CICS_HOME/lib

# Build the correct classpath
export BUILT_CP=$CICS_HOME/utils/namespace/dfhnsutils.jar
export BUILT_CP=$BUILT_CP:$CLIB/security/dfhreg.jar
export BUILT_CP=$BUILT_CP:$CLIB/dfjname.jar:$CLIB/websphere.jar
export BUILT_CP=$BUILT_CP:$CLIB/dfjcicsras.jar
export BUILT_CP=$BUILT_CP:$CLIB/ras.jar:$CLIB/dfjwrap.jar

java -cp £BUILT_CP com.ibm.cics.naming.utils.DFHBuildSNS -Xmx5M £"@"
```

This script has been affected by APARs, so you should change it to the above example to ensure it works!

I created another script (`d2`) to run the `MyDFHBuildSNS` script (which calls the java class):

```
MyDFHBuildSNS                                                    \
 -ldapserver  ldap://winmvs2c.hursley.ibm.com:2389               \
 -principal   "cn=admin"                                         \
 -credentials secret                                             \
 -containerdn "ibm-wsnTree=CicsTest,ou=RAH,o=IBM Hursley,C=UK" \
 -domain      SysProg
```

Figure 9: Specifying the Domain

The chosen name is supplied in the `-domain` parameter. The Anchor point dn is specified in the `-containerdn` field. (You defined this in “Creating the Tree anchor” on page 27.)

Observe the different syntax from `ldapmodify` and that the `-containerdn` parameter has to be enclosed in double quotes.

When the `d2` script is run, you should get the following output which shows that everything has worked correctly:

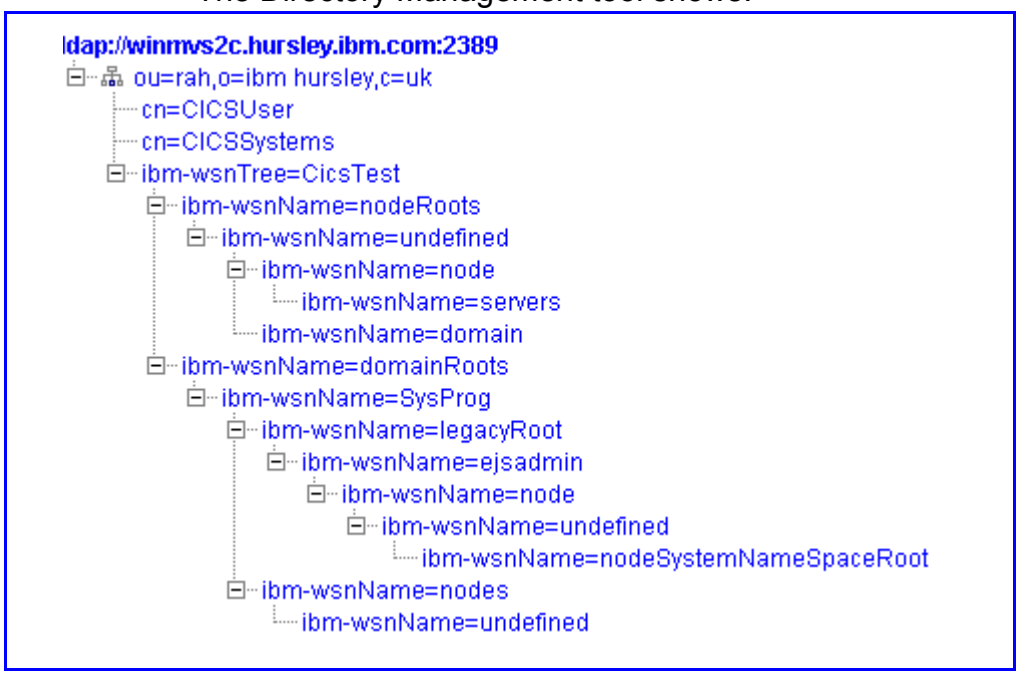
```
Processing request to build the system namespace:
LDAP Server: ldap://winmvs2c.hursley.ibm.com:2389
       Node: undefined
     Domain: SysProg
ContainerDN: ibm-wsnTree=CicsTest,ou=RAH,o=IBM Hursley,C=UK
  Principal: cn=admin

Checking current namespace structure.
Building the system namespace.
System namespace now ready for use by CICS TS.
-------------------------------------------------------------
```

Checking the Tree Structure

Once the script has run, there will be lots of additional entries in the LDAP Server which can be displayed. These entries were added under the supplied `ibm-wsnTree` anchor point which was set to `CicsTest`.

The LDAP Browser LHS panel will show:

The Directory Management tool shows:



You now need to create an entry for each Development CICS region within this structure. Goto “Adding the CICS region” on page 32.

Why you need to do this

This document is aimed at the Development/Test environment, so entries are going to be unique on a CICS region basis. Thus, each CICS region needs to be defined in the LDAP Structure. This definition is called a LDAP Subcontext.

I am going to impose the standard that the Subcontext name is going to be the Applid of a CICS region. (Each Development region does not share anything, so they need individual entries in the LDAP Server.) In the Production environment, different criteria will apply.

The name chosen for the Subcontext is used in the RDO CORBASERVER definition for the JNDIPREFIX field. This name is case sensitive. (See “RDO CORBASERVER” on page 46.)

Repeat the actions in this section for all required CICS regions (change the red items in Figure 10, ‘Creating the Subcontext/JNDIPrefix,’ on page 34).

Creating the ldif file

The Applid of my CICS region is IYCKRAH6, so this is what I am going to use as my Subcontext name and consequently use in all RDO CorbaServer JNDIPREFIX entries within the CSD for that region.

The subcontext is created via the ldapmodify utility. Some commands are in the `/usr/lpp/cicsts/cicsts22/utils/namespace/dfhNewCICSSubcontext.ldif` file. Copy this as `MydfhNewCICSSubcontext.ldif`. It should look like:

```
dn: ibm-wsnName=iycwabcd,
    ibm-wsnName=legacyRoot,
    ibm-wsnName=PLEX2,
    ibm-wsnName=domainRoots,
    ibm-wsnTree=t1,
    o=WASNaming,
    c=us
ibm-wsnname: iycwabcd
javaclassname: com.ibm.ws.naming.ldap.WsnLdapContextImpl
ibm-wsnentrytype: PrimaryContext
ibm-wsnnametreecontainerdn: ibm-wsnTree=t1,
        o=WASNaming,
        c=us
objectclass: ibm-wsnEntry
objectclass: ibm-wsnPrimaryContextLocation
ibm-wsnpathfromcontainer: ibm-wsnName=iycwabcd,
        ibm-wsnName=legacyRoot,
        ibm-wsnName=PLEX2,
        ibm-wsnName=domainRoots
aclentry: access-id:cn=CICSUser,c=US:object:ad:normal:rwsc
aclentry: group:CN=ANYBODY:normal:rsc
aclentry: access-id:cn=CICSSystems,c=US:object:ad:normal:rwsc
```

You must modify most of this file to use your assigned name.

It should end up like (ensure that trailing commas and colons are not omitted):

```
dn: ibm-wsnName=IYCKRAH6,
    ibm-wsnName=legacyRoot,
    ibm-wsnName=SysProg,
    ibm-wsnName=domainRoots,
    ibm-wsnTree=CicsTest,
    ou=RAH,
    o=IBM Hursley,
    c=UK
entryOwner: access-id:cn=admin
entryOwner: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK
ibm-wsnname: IYCKRAH6
javaclassname: com.ibm.ws.naming.ldap.WsnLdapContextImpl
ibm-wsnentrytype: PrimaryContext
ibm-wsnnametreecontainerdn: ibm-wsnTree=CicsTest,
                            ou=RAH,
                            o=IBM Hursley,
                            c=UK
objectclass: ibm-wsnEntry
objectclass: ibm-wsnPrimaryContextLocation
ibm-wsnpathfromcontainer: ibm-wsnName=IYCKRAH6,
                          ibm-wsnName=legacyRoot,
                          ibm-wsnName=SysProg,
                          ibm-wsnName=domainRoots
aclentry: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK:
          object:ad:normal:rwsc
aclentry: group:CN=ANYBODY:normal:rsc
aclentry: access-id:cn=CICSSystems, ou=RAH, o=IBM Hursley, c=UK:
          object:ad:normal:rwsc
```

Figure 10: Creating the Subcontext/JNDIPrefix

The red items are the Subcontext/JNDIPREFIX name which is case sensitive. The blue items are the domain (see Figure 9, ‘Specifying the Domain,’ on page 30). The green items are the anchor points (see Figure 8, ‘Creating the WebSphere Tree anchor,’ on page 27). The magenta items are the suffix (see Figure 6, ‘Creating the Suffix,’ on page 18).

The case-sensitive Userids (see Figure 7, ‘Adding CICS userids,’ on page 23) are given write access to this Subcontext as shown.

Observe that there are two `entryOwner` entries: the `cn=admin` one should correspond to the LDAP Server id (which is used in all the ldap commands). The explicit addition of this entry permits administrator access through the Directory tool.

The second `entryOwner` names the userid that is going to be responsible for the Subcontext, namely that used by the CICS region (see Section ”LDAP access Userid (java.naming.security.principal)” on page 43).

The Subcontext is created via ldapadd:

```
ldapadd     -v                                    \
            -h winmvs2c.hursley.ibm.com -p 2389   \
            -D "cn=admin"               -w secret \
            -f /u/rharri1/MydfhNewCICSSubcontext.ldif
```

Checking the results

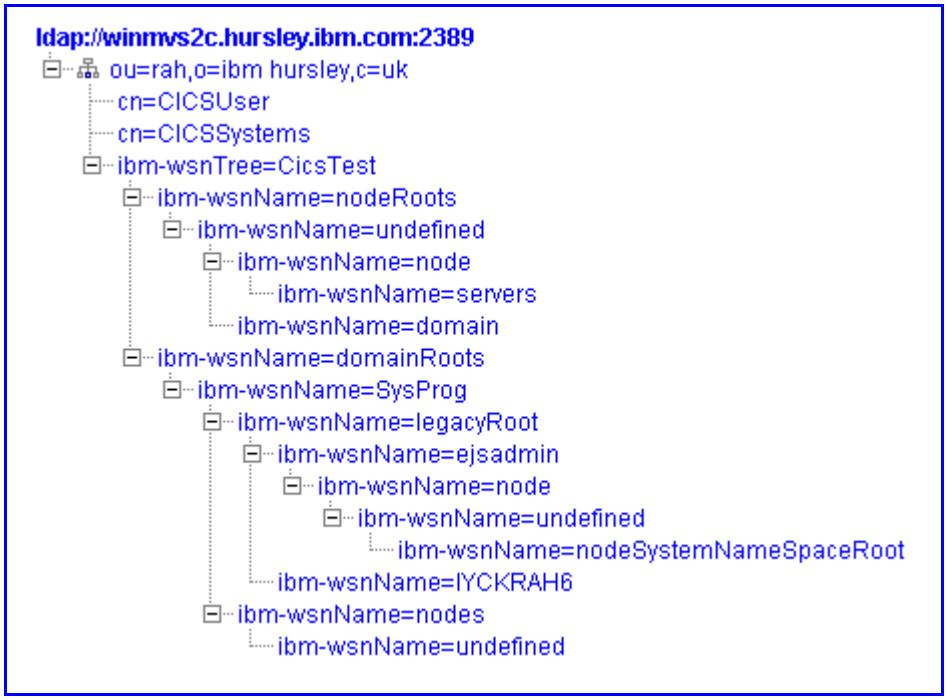
Using the tools

The LDAP Browser (after refreshing) will show the new SubContext:

Name	Value	Type	Size
ibm-wsnname	IYCKRAH6	text attribute	8
javaclassname	com.ibm.ws.naming.ldap.WsnLdapContextImpl	text attribute	41
ibm-wsnentrytype	PrimaryContext	text attribute	14
ibm-wsnnametreec...	ibm-wsnTree=CicsTest, ou=RAH, o=...	text attribute	127
objectclass	ibm-wsnEntry	text attribute	12
objectclass	ibm-wsnPrimaryContextLocation	text attribute	29
objectclass	TOP	text attribute	3
ibm-wsnpathfromc...	ibm-wsnName=IYCKRAH6, ibm-wsnName=legacyRoot,...	text attribute	162
createtimestamp	20020328145000.364201Z	operational attribute	22
modifytimestamp	20020328145000.364201Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Note that the `JavaClassName` attribute has appeared which is set to an IBM-supplied java class.

The Directory Management tool also shows the new entry:



but it can also show the access set on it:

Subject			Granted rights						
Remove	Distinguished name	Type	Add	Delete	Class	Read	Write	Search	Compare
☐	cn=anybody	group	☐	☐	Normal	☑	☐	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐
☐	cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK :	access-id	☑	☑	Normal	☑	☑	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐
☐	cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK :	access-id	☑	☑	Normal	☑	☑	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐

You can see that both the CICS Userids can manage the subcontext (put items into it and take items from it).

Using commands

You can issue this command:

```
ldapsearch                                                  \
            -h winmvs2c.hursley.ibm.com -p 2389             \
            -D "cn=admin"                  -w secret        \
            -b "ou=rah,o=ibm hursley,c=uk"                  \
               "(|(ibm-wsnName=IYCKRAH6))"                  \
               aclentry aclpropogate aclsource              \
               entryOwner ownerpropagate ownersource
```

which should yield something like (some lines split for readability):

```
ibm-wsnName=IYCKRAH6,  ibm-wsnName=legacyRoot,
ibm-wsnName=SysProg,   ibm-wsnName=domainRoots,
ibm-wsnTree=CicsTest,   ou=RAH,   o=IBM Hursley,   c=UK
entryowner=access-id:cn=admin
entryowner=access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK
aclentry=group:cn=anybody:normal:rsc
aclentry=access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK :
         object:ad:normal:rwsc
aclentry=access-id:cn=CICSSystems,ou=RAH,o=IBMHursley,c=UK :
         object:ad:normal:rwsc
ownerpropagate=TRUE
aclsource=ibm-wsnName=IYCKRAH6,   ibm-wsnName=legacyRoot,
   ibm-wsnName=SysProg,           ibm-wsnName=domainRoots,
   ibm-wsnTree=CicsTest,   ou=RAH,   o=IBM Hursley, c=UK
ownersource=ibm-wsnName=IYCKRAH6,  ibm-wsnName=legacyRoot,
            ibm-wsnName=SysProg,   ibm-wsnName=domainRoots,
            ibm-wsnTree=CicsTest,   ou=RAH,   o=IBM Hursley,c=UK
```

What to do next

You have finished the configuration of the LDAP Server to use by CICS!

Continue by configuring the java-related parts of CICS. This is described in “CICS relationships” on page 42. After that define a CorbaServer for use as shown in “CorbaServers” on page 47 which will enable you to test things out.

However, depending upon whether or not a bug has been fixed, you may need to do another ldif operation as described in “Avoiding the CICS Retraction bug” on page 39. I suggest you do not do this unless the bug is present. It appears in the circumstances described in “What is the bug?” on page 39.

What is the bug?

There is, ahem, a bit of an, err, bug in the way CICS deletes entries from the LDAP Hierarchy.

CICS will over enthusiastically delete the SubContext level if it is empty: so deleting all the Security settings described in “Adding the CICS region” on page 32.

You can end up, after issuing CICS commands, with a structure that omits the SubContext (the IYCKRAH6 level in my case):

Circumventing the bug

The easiest way to circumvent the bug is to ensure that the SubContext level never becomes empty. The simplest way to do this is to create a fake user.

I have coded up an ldif file called `Myfix.ldif` containing a dummy entry:

```
dn: cn=Fix to prevent CICS from deleting this level,
    ibm-wsnName=IYCKRAH6,
    ibm-wsnName=legacyRoot,
    ibm-wsnName=SysProg,
    ibm-wsnName=domainRoots,
    ibm-wsnTree=CicsTest,
    ou=RAH,
    o=IBM Hursley,
    c=UK
changetype: add
objectclass: person
cn: Fix to prevent CICS from deleting this level
sn: Fake entry
userPassword: secret
entryOwner: access-id:cn=admin
aclentry: group:cn=anybody:normal:rsc
aclentry: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK :
          normal:rsc
aclentry: access-id:cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK :
          normal:rsc
```

And invoked it via a:

```
ldapadd    -v                                    \
           -h winmvs2c.hursley.ibm.com -p 2389   \
           -D "cn=admin"               -w secret \
           -f Myfix.ldif
```

The crucial part is in red - it's the level of the SubContext that CICS will erroneously remove if given the chance.

In subsequent chapters of this document, this fake entry is not shown to avoid confusion (and irrelevance once the bug is fixed!).

After running the command, the LDAP Browser will show:

The Directory Tool additionally shows that the LDAP User associated with CICS cannot manipulate it:

Subject		Granted rights					
Distinguished name	Type	Add	Delete	Class	Read	Write	Search
cn=anybody	group	☐	☐	Normal	☑	☐	☑
				Sensitive	☐	☐	☐
				Critical	☐	☐	☐
cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK:	access-id	☐	☐	Normal	☑	☐	☑
				Sensitive	☐	☐	☐
				Critical	☐	☐	☐
cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK:	access-id	☐	☐	Normal	☑	☐	☑
				Sensitive	☐	☐	☐
				Critical	☐	☐	☐

JVM System Properties

JVM System Properties are set in an HFS file named via a member in the DFHJVM PDS via the `JVMPROPS` setting (conventionally called the `system.properties` file). The member used is set on the RDO PROGRAM definition if using a normal Java class (one which is executed via a `main` method) or determined from a matching RDO REQUESTMODEL if runing an Enterprise Bean. See the *CICS Java* and *CICS RDO* books for details.

CICS will use a fixed member name of `DFHJVMPR` for its operations, so this member must exist in the `DFHJVM` PDS and contain, amongst other things, the following entries.

Some of these properties relate to the LDAP Server and so are taken directly from the configuration of the LDAP Server.

In the settings that follow, the line breaks are for readability.

Nameserver (`com.ibm.cics.ejs.nameserver`)

This is the IP name of the LDAP server including the port number used for access. (See Figure 3, 'LDAP initial configuration file,' on page 8.)

In my case, the setting is:

```
com.ibm.cics.ejs.nameserver=ldap://winmvs2c.hursley.ibm.com:2389
```

Container Distinguished Name (`com.ibm.ws.naming.ldap.containerdn`)

This is the name of the anchor point for the configuration. (See Figure 8, 'Creating the WebSphere Tree anchor,' on page 27.)

In my case, the setting is:

```
com.ibm.ws.naming.ldap.containerdn=
        ibm-wsnTree=CicsTest,ou=RAH,o=IBM Hursley,C=UK
```

Anchor point (com.ibm.ws.naming.ldap.noderootrdn)

This is the name under which all associated entries are placed into the LDAP structure. You do not specify the full dn, only the bit after the containerdn. Recall that dns are specified in a left-to-right fashion with the top of the tree being the last rdn. (See Figure 9, 'Specifying the Domain,' on page 30.) This setting appears a little odd in that all three entries have the same name (fixed by the WebSphere naming schema), and that only the middle one is variable.

In my case, the setting is:

```
com.ibm.ws.naming.ldap.noderootrdn=
        ibm-wsnName=legacyRoot,
        ibm-wsnName=SysProg,
        ibm-wsnName=domainRoots
```

LDAP access Userid (java.naming.security.principal)

This is the Userid used by CICS to access the LDAP Server (see "Creating the Users" on page 23). In my case the setting is:

```
java.naming.security.principal=
        cn=cicsuser,ou=RAH,o=IBM Hursley,c=UK
```

LDAP access Password (java.naming.security.credentials)

This is the Password for the Userid used by CICS to access the LDAP Server (see "Creating the Users" on page 23). In my case the setting is:

```
java.naming.security.credentials=secret
```

Java Security Mechanism (java.naming.security.authentication)

The Java Security mechanism that supports the LDAP access Userid is governed by a fixed setting for this attribute of `simple`.

```
java.naming.security.authentication=simple
```

JNDI constructor class (java.naming.factory)

This entry is fixed as it contains the java class used to manipulate the LDAP Server. This fixed entry is:

```
java.naming.factory.initial=
        com.ibm.sphere.naming.WsnInitialContextFactory
```

My System.properties file

The `system.properties` file that I am using looks like:

```
#
# Host LDAP
#
java.naming.factory.initial=com.ibm.sphere.naming.WsnInitialContextFactory
com.ibm.cics.ejs.nameserver=ldap://winmvs2c.hursley.ibm.com:2389
com.ibm.ws.naming.ldap.containerdn=ibm-wsnTree=CicsTest,ou=RAH,o=IBM Hursley,C=UK
com.ibm.ws.naming.ldap.noderootrdn=ibm-wsnName=legacyRoot,ibm-wsnName=SysProg,ibm-wsnName=domainRoots
java.naming.security.authentication=simple
java.naming.security.principal=cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK
java.naming.security.credentials=secret
```

Figure 11: System.Properties file

RDO CORBASERVER

The `JNDIPREFIX` for **all** CorbaServer objects in the CICS region should be set to the case sensitive SubContext name (see Figure 10, 'Creating the Subcontext/JNDIPrefix,' on page 34). As this is case sensitive, switch the terminal into mixed mode input via `CEOT UC` before doing `CEDA` (which you will have to specify in UPPERCASE!).

The JNDIPrefix (and SubContext name) is `IYCKRAH6` in my case.

You have the choice of exposing a second-level name after the SubContext name by quoting the JNDIPREFIX as `IYCKRAH6/name`.

Examples of both types are given in "CorbaServers used" on page 47.

In fact, you can have many /s in the JNDIPREFIX: you just get lots of extra levels in the LDAP hierarchy. However, this configuration is not recommended.

Introduction

This section shows how the LDAP Server should react when Publishing and Retracting CorbaServer definitions. Publishing means putting the GenericFactory object for the CorbaServer into the LDAP hierarchy. Retracting means deleting it along with any Bean-related information.

This chapter only discusses CorbaServers from the LDAP perspective.

A GenericFactory is used to locate the Home Interface for an Enterprise Bean. It is inserted into the LDAP structure as a type of `corbaIor` (InterOperableResource for a Corba Object) and contains addressing information. It looks like:

```
corbaior        IOR:000000000000002c49444c3a6f6d672e6f72672f436f734c6966654379636c65...
```

Figure 12: Initial part of an IOR

TCPIPService definitions

Each of the CorbaServers has a separate TCPIPSERVICE definition installed (not relevant to a LDAP discussion).

CorbaServers used

In order to show what happens in the LDAP Server for the Publication and Retraction of a CorbaServer, I am going to use two RDO-defined CORBASERVER objects (recall that JNDIPREFIX is case sensitive, use `CEOT UC` to put your terminal into mixed-case mode):

C000 just has the SubContext name (IYCKRAH6 in my case) as the JNDIPREFIX.

```
 CORbaserver     : C000
 Group           : RAHEJ
 DEscription     : CORBASERVER C000
 Jndiprefix      : IYCKRAH6
 Autopublish     : No                  Yes | No
 SEssbeantime    : 00 , 00 , 10        0-99 (Days,Hours,Mins)
 SHelf           : /u/rharri1/shelf
 DJardir         :
SERVER ORB ATTRIBUTES
 Host            : IYCKRAH6.C000
                 :
                 :
                 :
                 :
CLIENT ORB ATTRIBUTES
 CErtificate     :
TCPIP SERVICES
 Unauth          : TCC000
 CLientcert      :
 SSLUnauth       :
```

C0001 has the SubContext name and the name of the CorbaServer exposed (IYCKRAH6/C001):

```
 CORbaserver     : C001
 Group           : RAHEJ
 DEscription     : CORBASERVER C000
 Jndiprefix      : IYCKRAH6
 Autopublish     : No                  Yes | No
 SEssbeantime    : 00 , 00 , 10        0-99 (Days,Hours,Mins)
 SHelf           : /u/rharri1/shelf
 DJardir         :
SERVER ORB ATTRIBUTES
 Host            : IYCKRAH6.C001
                 :
                 :
                 :
                 :
CLIENT ORB ATTRIBUTES
 CErtificate     :
TCPIP SERVICES
 Unauth          : TCC001
 CLientcert      :
 SSLUnauth       :
```

CEMT Shows that they have been correctly installed:

```
  I CORB
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Corba(C000) Inser       Sessb( 000010 ) Unaut(TCC000  )
  Corba(C001) Inser       Sessb( 000010 ) Unaut(TCC001  )
```

Initial LDAP Hierarchy

Browser display

The initial view of the LDAP hierarchy for the LDAP Browser is:

Name	Value	Type	Size
ibm-wsnname	IYCKRAH6	text attribute	8
javaclassname	com.ibm.ws.naming.ldap.WsnLdapContextImpl	text attribute	41
ibm-wsnentrytype	PrimaryContext	text attribute	14
ibm-wsnnametreec...	ibm-wsnTree=CicsTest, ou=RAH, o=IBM Hursley,...	text attribute	127
objectclass	ibm-wsnEntry	text attribute	12
objectclass	ibm-wsnPrimaryContextLocation	text attribute	29
objectclass	TOP	text attribute	3
ibm-wsnpathfromc...	ibm-wsnName=IYCKRAH6, ibm-wsnName=legacyRoot, ...	text attribute	162
createtimestamp	20020402110725.314802Z	operational attribute	22
modifytimestamp	20020402110725.314802Z	operational attribute	22
modifiersname	CN=ADMIN	operational attribute	8
creatorsname	CN=ADMIN	operational attribute	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Figure 13: Initial LDAP Hierarchy: LDAP Browser

Directory Tool display

The initial hierarchy a seen through the Directory Management tool is:

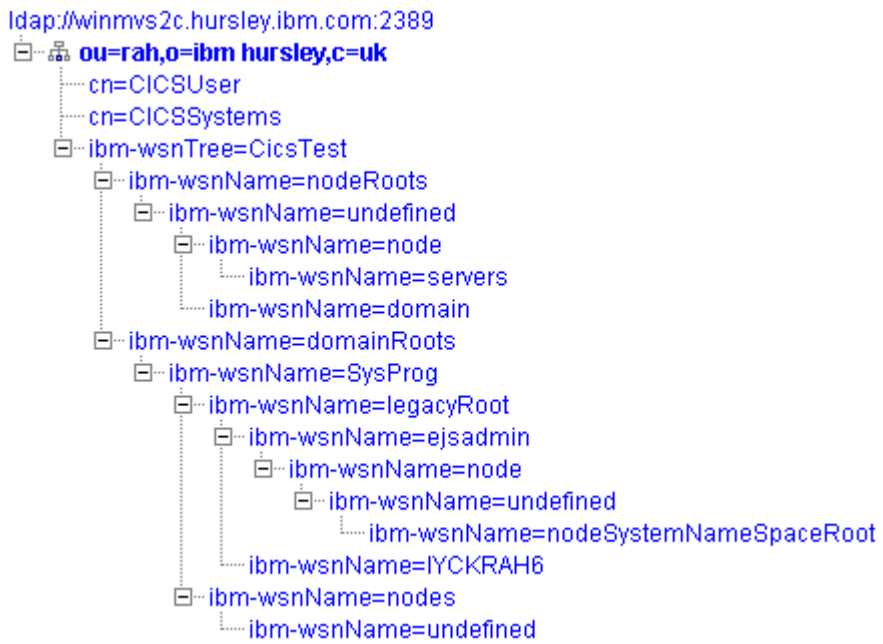


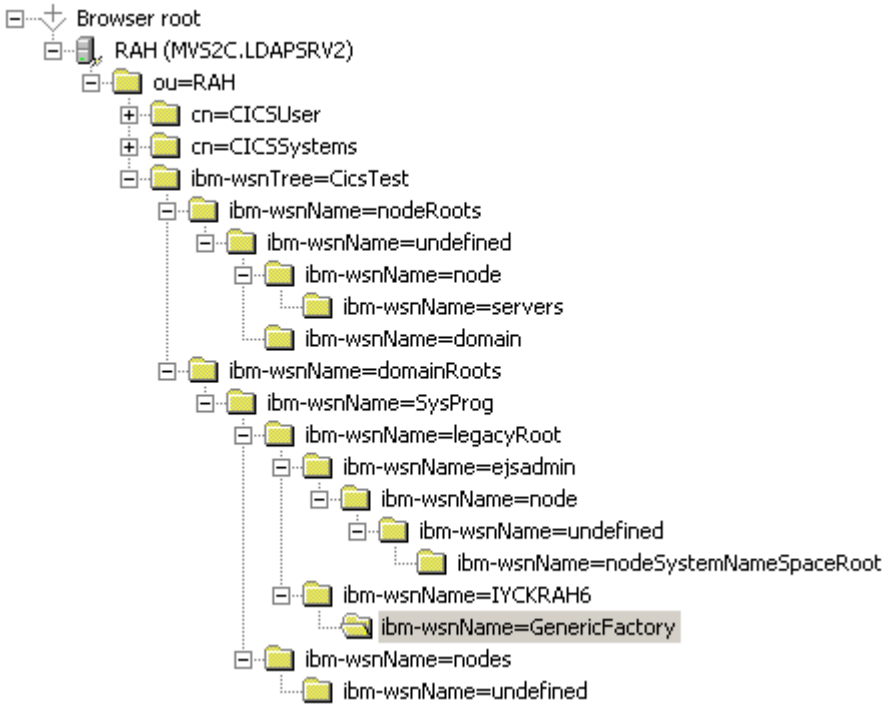
Figure 14: Initial LDAP Hierarchy: Directory Tool

Results of Publishing the CorbaServer

JNDIPrefix without a /

If one does a `CEMT PERFORM CORBA(C000) PUBLISH` to insert information into the LDAP Server, this inserts the GenericFactory into the Hierarchy.

After rebinding, the new GenericFactory entry can be seen showing a type of `CORBAOBJECT`:



Name	Value	Type	Size
ibm-wsnname	GenericFactory	text attribute	14
ibm-wsnentrytype	IORLeaf	text attribute	7
corbaior	IOR:000000000000002c49444c3a6f6d672e6f72672f436f734c6966654379636c65...	text attribute	508
objectclass	ibm-wsnEntry	text attribute	12
objectclass	corbaObjectReference	text attribute	20
objectclass	CORBAOBJECT	text attribute	11
objectclass	TOP	text attribute	3
javaclassname	com.ibm.cics.iiop.cso._GenericFactoryImpl	text attribute	41
createtimestamp	20020402113347.590634Z	operational attribute	22
modifytimestamp	20020402113347.590634Z	operational attribute	22
modifiersname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	37
creatorsname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	37
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Figure 15: Publication result for JNDIPREFIX without a / : LDAP Browser

The Directory tool shows:

Figure 16: Publication result for JNDIPREFIX without a / : Directory Tool

JNDIPrefix with a /

If one does a `CEMT PERFORM CORBA(C001) PUBLISH` to insert information into the LDAP Server, this inserts the GenericFactory into the Hierarchy under a lower-level name (which is the part after the JNDIPREFIX /).

The Browser shows the intermediate level (the bit after the /) as just another LDAP context:

Name	Value	Type	Size
ibm-wsnName	GenericFactory	entry	616
ibm-wsnnametreec...	ibm-wsnTree=CicsTest,ou=RAH,o=IBM Hursley,C=UK	text attribute	46
ibm-wsnname	CSC001	text attribute	6
ibm-wsnentrytype	PrimaryContext	text attribute	14
objectclass	ibm-wsnEntry	text attribute	12
objectclass	ibm-wsnPrimaryContextLocation	text attribute	29
objectclass	TOP	text attribute	3
ibm-wsnpathfromc...	ibm-wsnName=CSC001,ibm-wsnName=IYCKRAH6,ibm-wsnName=legacyRoot,ibm...	text attribute	106
javaclassname	javax.naming.Context	text attribute	20
createtimestamp	20020402114513.460372Z	operational attribute	22
modifytimestamp	20020402114513.460372Z	operational attribute	22
modifiersname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	37
creatorsname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	37
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Figure 17: Publication result for JNDIPREFIX with a / : LDAP Browser

The Management tool shows:

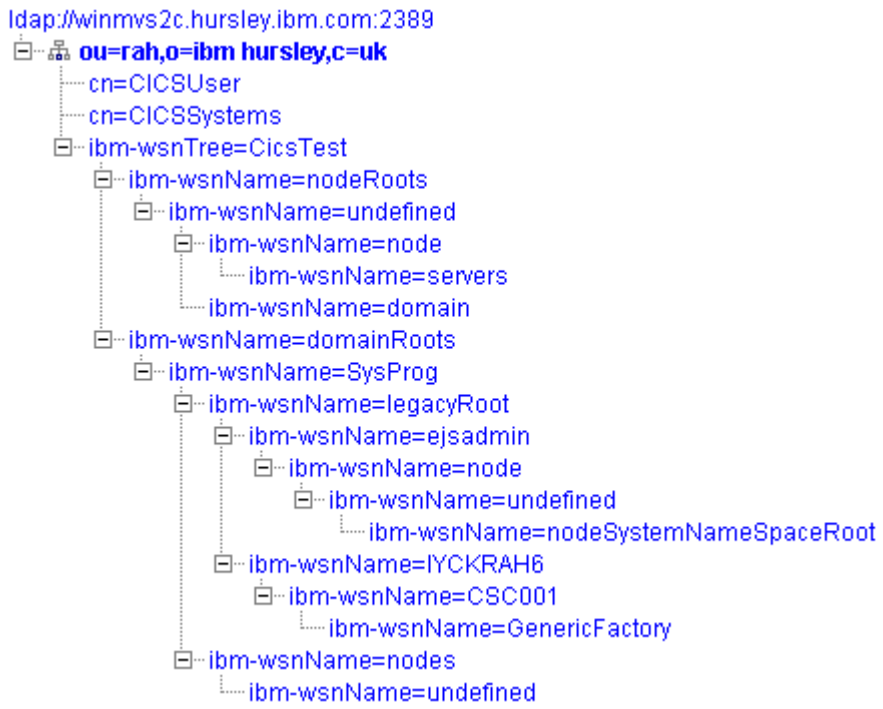


Figure 18: Publication result for JNDIPREFIX with a / : Directory Tool

Retracting a Corbaserver

The opposite of Publishing a CorbaServer is to retract it. This removes the IOR from the LDAP Server and so makes the CorbaServer, and all the Beans within it, unavailable for use.

This is done via a `CEMT PERFORM CORB(xxxx) RETRACT` command.

After issuing a `CEMT P CORB(C001) RETRACT`, the situation shown in Figure 18, 'Publication result for JNDIPREFIX with a / : Directory Tool,' on page 54 will return to Figure 14, 'Initial LDAP Hierarchy: Directory Tool,' on page 50.

And the equivalent `CEMT P CORB(C000) RET` will return from Figure 15, 'Publication result for JNDIPREFIX without a / : LDAP Browser,' on page 51 to Figure 13, 'Initial LDAP Hierarchy: LDAP Browser,' on page 49.

A CICS Bug

There is, ahem, a CICS bug, err, that rather enthusiastically deletes the lowest level in the LDAP Hierarchy when it is empty.

Consequently, after a Retraction, the LDAP Structure may end up like:

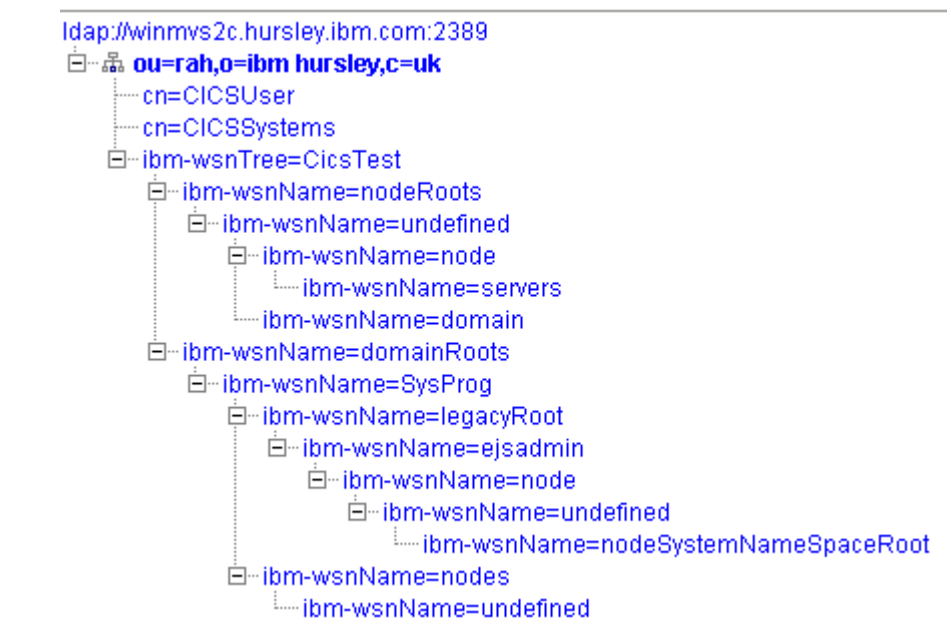


Figure 19: CICS Retraction bug

If this happens, do some more configuration as discussed in "Circumventing the bug" on page 40.

CorbaServers own DJars

A DJar (Deployed Jar file) contains Enterprise Bean code. In CICS terminology, a RDO DJAR definition just contains the name of the HFS jar file and which CorbaServer into which the Beans contained within the jar file are to be placed.

Before a Bean can be used by a client, it has, like a CorbaServer, to be published to the LDAP Server (from whence a client obtains the addressing information). Like the CorbaServer, this publication involves putting the Bean IOR into the LDAP Hierarchy under the owning CorbaServer.

Publishing a DJar

Here is a RDO definition for a DJAR:

```
DJar            : C001D001
Group           : RAHEJ
Description     : HELLOWORLD
Corbaserver     : C001
Hfsfile         : /u/rharri1/HelloWorldEJB.jar
```

The name of the RDO DJAR object itself is somewhat irrelevant. It's only a mechanism for associating the HFS name of the Deployed jar file (specified in mixed case) and the owning CorbaServer.

CICS has various mechanisms for creating DJar definitions, but these are outside of the scope of this document.

If a CorbaServer already has installed DJAR RDO definitions active upon Publication, then the DJARs are also published. Similarly, the Retraction of a CorbaServer will retract all associated DJars.

However, individual DJars can themselves be Published and Retracted and this is what this chapter is considering. In fact, it is not the DJar that is being Published or Retracted but definitions of all the Enterprise Beans within the relevant jar file.

The example in this Chapter is using the CICS EJB `HelloWorld` sample.

Publishing using CEMT

A `CEMT I DJAR` command shows installed DJars (only one in my case):

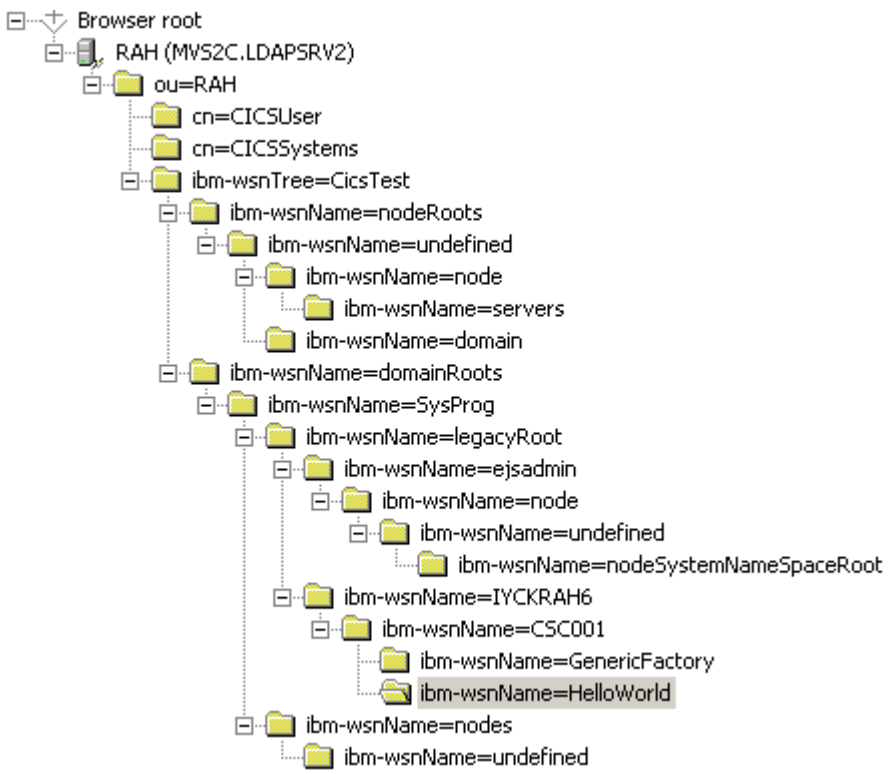
```
I DJAR
STATUS:  RESULTS
 Djar(C001D001                        ) Corba(C001) Inser
    Dates(20020403) Times(13:05:00) Hfsfi(/u/rharri1/HelloWorldEJB.j)
```

Note that the HFS file name is truncated on the display.

I published this individual DJAR via a `CEMT PERFORM DJAR(C001D001) PUBLISH` command. If the CorbaServer was being Published when this DJar definition was active, it would have been Published along with the CorbaServer.

LDAP results of DJar publication

The LDAP Browser shows the addition of the Bean under the CorbaServer entry:

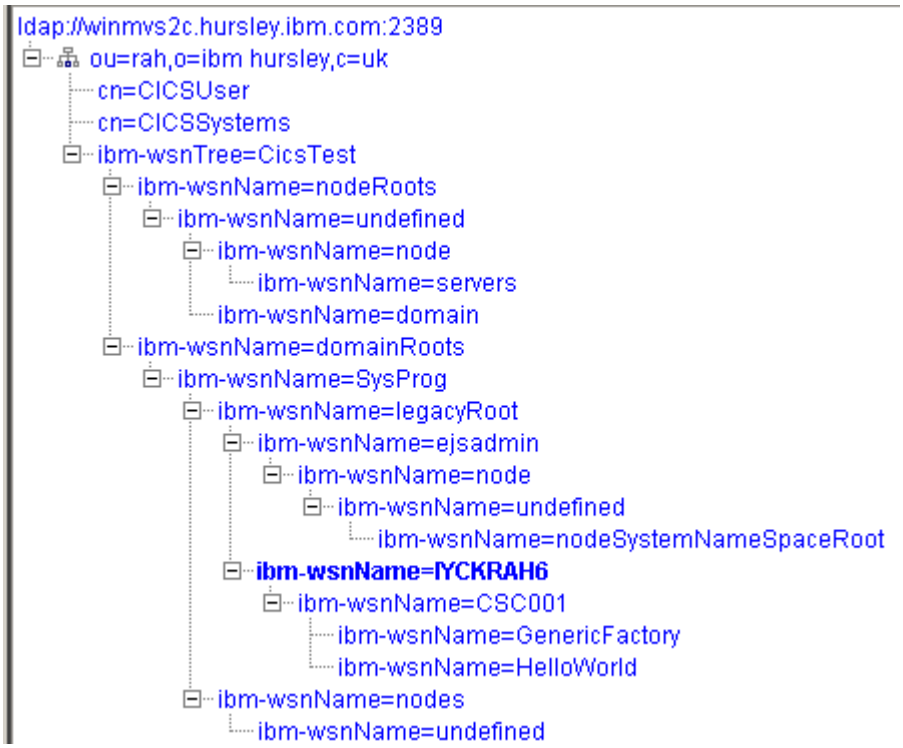


Name	Value	Type	Size
ibm-wsnname	HelloWorld	text attribute	10
ibm-wsnentrytype	IORLeaf	text attribute	7
corbaior	IOR:000000000000000030524d493a636963732e73616d706c652e48656c6c6f576f7...	text attribute	884
objectclass	ibm-wsnEntry	text attribute	12
objectclass	corbaObjectReference	text attribute	20
objectclass	CORBAOBJECT	text attribute	11
objectclass	TOP	text attribute	3
javaclassname	cics.sample.EJSRemoteStatelessHelloWorldHome	text attribute	44
createtimestamp	20020403131034.322953Z	operational attribute	22
modifytimestamp	20020403131034.322953Z	operational attribute	22
modifiersname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	37
creatorsname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	37
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	operational attribute	35

Figure 20: Publication result for DJar : LDAP Browser

You can see that the Bean has an IOR for access, and that the `javaclassname` entry names the Home Interface for the Bean. Unlike most of the other parts of the LDAP Hierarchy, you can see that CICS has created the entry.

The Directory Tool also shows the security information:



Subject			Granted rights						
Remove	Distinguished name	Type	Add	Delete	Class	Read	Write	Search	Compare
☐	cn=anybody	group	☐	☐	Normal	☑	☐	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐
☐	cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK:	access-id	☑	☑	Normal	☑	☑	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐
☐	cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK:	access-id	☑	☑	Normal	☑	☑	☑	☑
					Sensitive	☐	☐	☐	☐
					Critical	☐	☐	☐	☐

Figure 21: Publication result for DJar : Directory Tool

Observe that it has inherited access from the owning CorbaServer entry.

Retracting the DJAR

The DJAR is Retracted, removing its IOR from the LDAP Server, via a `CEMT P DJAR(C001D001) RET` command. All DJars associated with a CorbaServer are retracted if the owning CorbaServer is itself Retracted.

Checking Spellings

Initial problems with using a LDAP Server will probably arise from the specification of the `system.properties` file.

The file being used is named in the `DFHJVMPR` member of the DFHJVM PDS.

Ensure that the spellings are correct! As most things are case sensitive, an unlikely lower-case letter may spell DiSasTer.

Deleting the configuration

Of course, you have diligently followed everything in this document down to the last comma and colon - so things will work first time!

In the *unlikely*[1] event that you need to delete everything and restart, the easiest way is to use the Directory Tool and delete the top-level (suffix) item.

Alternatively, use the following script file to do the deletion. The complexity of it arises from the fact that `ldapsearch` lists things in hierarchy order, but items have to be deleted from the bottom up. Another factor is that awk arrays use [square brackets] and this can cause code-page problems.

I've called the script file `delallldap` (remember to `chmod a+rwx` it and, maybe, changing `£`s to `$`s etc. together with the apt namings):

1. In other words, all the time until you get things working!

```
#
# Shell script to delete everything from schema downwards
#

pserver="winmvs2c.hursley.ibm.com"
pport="2389"
puserid="cn=admin"
ppassword="secret"
pschema="ou=RAH,o=IBM Hursley,c=UK"

echo
echo " --Going to delete-- "
echo

ldapsearch -h £pserver  -p £pport      \
           -D £puserid  -w £ppassword  \
           -b "£pschema"               \
              "(|(objectclass=*))"     \
              dn                      |\
 cat

echo
echo " --Deleting (in reverse order)-- "
echo

ldapsearch -h £pserver  -p £pport      \
           -D £puserid  -w £ppassword  \
           -b "£pschema"               \
              "(|(objectclass=*))"     \
              dn                      |\
                                                  \
    awk 'BEGIN { FS = " " ; RS = "" ; recs = "" }     \
                                                  \
               {                                  \
                 if ( NR == 1 ) recs = £0 ;       \
                          else  recs = £0 "\001" recs \
               }                                  \
                                                  \
         END   {                                  \
                print recs                        \
               }                                  \
        '                                        |\
                                                 \
    awk 'BEGIN { FS = " " ; RS = "\001"         }  \
                                                 \
               {                                 \
                 print £0                        \
               }                                 \
        '                                       |\
                                          \
    ldapdelete -v                         \
               -h £pserver  -p £pport     \
               -D £puserid  -w £ppassword

echo
echo " --End of Deletion script-- "
echo
#
#
# End of Shell script
```

Figure 22: Shell script to delete all LDAP entries

CICS Tracing

Unfortunately, most of the function of CICS' LDAP processing is contained within Java code that does not have the level of tracing traditionally enjoyed by CICS functions. The best you can do is turn on `II` and `EJ` domain tracing, but this is not usually too helpful.

LDAP Server tracing

The LDAP server will accept MVS Modify commands to control tracing. The syntax, from SDF, is `/F jobname,APPL=DEBUG=n` where n is a tracing level number. To turn LDAP tracing off use `/F jobname,APPL=DEBUG=0`,

The level numbers are documented in the *LDAP Admin* book, but `DEBUG=133` is the most useful setting as this shows security (acl) processing as well as the routines used by the Server.

CICS Messages

The messages that CICS outputs for LDAP Processing are constrained because they come from the aforementioned java code, and so are not under CICS' control. In general, they will contain the level (rdn) in the hierarchy at which the error condition occurred, or at least part of the hierarchy passed from CICS at which an objection was detected. Investigate around this rdn to detect the problem.

LDAP Level mismatch

You should use the LDAP Browser or Directory tool to display the LDAP hierarchy and consider why the mismatch has arisen. This is easier said than done, but it should be obvious if there is a missing level.

Case Sensitivity

Always inspect the case of the request and compare with what is in the LDAP Server. Most things tend to be case sensitive, so this can commonly produce errors.

In the case of CorbaServer operations, the RDO definition for JNDIPREFIX is case sensitive, so if the terminal which created it was not in mixed mode (`CEOT UC`) then the LDAP SubContext must be in Upper Case (see “Adding the CICS region” on page 32). The solution is to use a terminal which has temporarily switched into mixed-mode input before altering the RDO CorbaServer entry.

Userid failures

If the Userid (see “LDAP access Userid (java.naming.security.principal)” on page 43) and/or the password (see “LDAP access Password (java.naming.security.credentials)” on page 43) is incorrect, this will be quickly apparent though a CICS message.

ACL violations

LDAP Security violations can arise if the Userid used by CICS for LDAP access (see “LDAP access Userid (java.naming.security.principal)” on page 43) is not authorised for the relevant LDAP hierarchy level. One of the causes for this is that you have not set the `entryowner` attributes correctly (see Figure 10, ‘Creating the Subcontext/JNDIPrefix,’ on page 34.

Introduction to JDBC 2.0 and DB2 on CICS

CICS has extensive facilities for accessing DB2 from traditional application programs. These have evolved over time and the latest supported DB2 is v7.1.

In the Java environment, access to a database is via Java Data Base Connectivity Version 2.0 protocols. JDBC 2.0 has evolved for an environment where a connection can be made to multiple databases and these connections have to be managed. This is called Connection Pooling.

JDBC 2.0 within CICS uses the underlying DB2 connection mechanisms provided by CICS (which are defined by RDO etc.) for application programs. The operational semantic of Connection Pooling (which is not visible to a java application) implied by the JDBC 2.0 protocols is not needed as CICS provides a superior (but equivalent) mechanism for optimising database connections.

JDBC 2.0 has the concept of direct connections to multiple databases (which is why they have to be managed). CICS' usage of DB2 has a different concept: a connection is always made to a single DB2 sub-system, and it is the responsibility of the DB2 instance to manage access to the required database.

The upshot of this is that the usage of JDBC 2.0 to access DB2 within the CICS environment is directly equivalent to that for application programs: a single DB2 is contacted and accessed.

Defining the DB2 database to be accessed

As CICS can only access a single DB2 instance, the java definition of it is simple. One should always define the connection so that the default URL is used for the database (as the underlying RDO-based mechanisms will correctly resolve it).

There are two ways of defining this (the first is preferred):

```
jdbc:default:connection

jdbc:db2os390sqlj:
```

These values can be placed in the `system.properties` file (see "JVM System Properties" on page 42) and resolved via a context lookup, or placed directly in the java object (not recommended).

Acquiring the DB2 Connection

Under JDBC 1.2 access was via the DriverManager Interface (which used the database URL directly). This technique does not require any JNDI or LDAP configuration.

Under JDBC 2.0 the preferred way of obtaining a database connection is via the DataSource interface. The DataSource interface uses JNDI operations to resolve a reference to a previously published object.

This published DataSource object is, essentially, empty, as it does not contain any meaningful information for access to DB2 from CICS. Consequently, it can be reused.

JDBC datatype for DB2 access

As CICS only accesses a single DB2 instance, the class required for the java Connection object is **`DB2SimpleDataSource`**.

Avoiding the JNDI function

If you are writing a java application **specifically** for the CICS environment, you do not need to bother about compatibility with the full JDBC 2.0 operational characteristics. You merely want to create the Connection Object and then use it. The intermediate step of populating the Connection Object can be omitted as there is nothing sensible with which to populate it. This has an **huge** performance benefit in avoiding processing associated with JNDI/LDAP operations.

The java code to do this would look like:

```
// Generate direct connection to DB2

DB2SimpleDataSource ds = new DB2DataSource() ;
Connection db2conn = ds.getConnection() ;

// Go and access DB2 source
```

Using JNDI lookup

If you have acquired the java database access code from an external source, or wish to write code with maximum portatability, you have to use a JNDI lookup to resolve the DB2 Database connection.

This section discusses this operation from the LDAP viewpoint.

Setting the JNDI key

By convention, the JNDI key used for JDBC access is of format:

```
jdbc/<database identity>
```

Consequently, you have to create a JNDI object with this required key. The lookup is going to be from within the JNDI environment provided by CICS. Thus, the item will be placed in the tree under the influences of the `Containerdn` (see "Container Distinguished Name (`com.ibm.ws.naming.ldap.containerdn`)" on page 42) and `Noderootdn` settings (see "Anchor point (`com.ibm.ws.naming.ldap.noderootrdn`)" on page 43).

In effect, you will be adding a JDBC leaf and, under that, entries for the names of the databases. I am going to call my object `IYCKRAH6/jdbc/CICSDB2instance` (as I am going to have my definition uniquely specified for my own CICS region).

You are quite at liberty to use any name you like for the database name, but it is a waste of time and effort to use more than one (as they all resolve to the same thing). Additionally, it does not matter what the JNDI object contains as CICS will ignore most of the settings as it already knows which DB2 it is going to contact.

Consequently, it is recommended that the context contains this `jdbc/<database>` name and so the use of a specific JNDI setting is avoided in the java code itself.

It is recommended that a key of `com.ibm.cics.datasource.name` is used for this lookup. Thus, `system.properties` would contain something like:

```
com.ibm.cics.datasource.name=IYCKRAH6/jdbc/CICSDB2instance
```

and be used via:

```
String contextDataBaseName = "com.ibm.cics.datasource.name" ;
String dataSourceName = System.getProperty(contextDataBaseName);
```

Resolving the Connection Object using JNDI

The name of the JNDI entry containing the connection object is then used to resolve the Connection before it is used to access the DB2 database:

```
Context ctx = new InitialContext() ;
DataSource ds = null ;
ds = lookupDataSource(ctx,dataSourceName) ;

Connection db2conn = ds.getConnection() ;

// Go and access DB2
```

Observe that using the JNDI method to resolve the Connection uses a `DataSource` object, whereas avoiding JNDI uses a `DB2SimpleDataSource` object.

The JNDI resolution step 'turns' the DataSource object into a DB2SimpleDataSource object for use in accessing the DB2 database. Strictly, this means that the DB2SimpleData class inherits from the DataSource class and so JDBC operations inherent in DataSource are implemented in DB2SimpleDataSource.

Publishing the Database Connection using LDAP

The object published to the LDAP server contains information necessary to alter the `DataSource` Object into a `DB2SimpleDataSource` object (so that DB2 can be accessed from within the Java environment within CICS).

LDAP definitions

You need to define to the LDAP server the correct information for the JNDI operation. This involves creating the `IYCKRAH6/jdbc/CICSDB2instance` entry in the correct place of the LDAP hierarchy (governed by the definitions used for the CICS region, which means everything upto and including the `IYCKRAH6` part is already present).

I have coded up a file called `Myjdbc.ldif` which contains the required definitions. The first part of this contains the definitions for the `jdbc` node:

```
# Define the JDBC 2.0 root

dn: ibm-wsnName=jdbc,
    ibm-wsnName=IYCKRAH6,
    ibm-wsnName=legacyRoot,
    ibm-wsnName=SysProg,
    ibm-wsnName=domainRoots,
    ibm-wsnTree=CicsTest,
    ou=RAH,
    o=IBM Hursley,
    c=UK
ibm-wsnName: jdbc
javaClassName: javax.naming.Context
ibm-wsnEntryType: PrimaryContext
ibm-wsnNameTreeContainerDN: ibm-wsnTree=CicsTest,
                            ou=RAH,
                            o=IBM Hursley,
                            c=UK
objectclass: ibm-wsnEntry
objectclass: ibm-wsnPrimaryContextLocation
ibm-wsnPathFromContainer: ibm-wsnName=jdbc,
                          ibm-wsnName=IYCKRAH6,
                          ibm-wsnName=legacyRoot,
                          ibm-wsnName=SysProg,
                          ibm-wsnName=domainRoots
entryOwner: access-id:cn=admin
entryOwner: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK
aclentry: group:cn=anybody:normal:rsc
aclentry: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK :
          object:ad:normal:rwsc
aclentry: access-id:cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK :
          object:ad:normal:rwsc
```

whilst the latter part contains the information for the JDBC accessed database:

```
# Define the JDBC 2.0 leaf node Data Source

dn: ibm-wsnName=CICSDB2instance,
    ibm-wsnName=jdbc,
    ibm-wsnName=IYCKRAH6,
    ibm-wsnName=legacyRoot,
    ibm-wsnName=SysProg,
    ibm-wsnName=domainRoots,
    ibm-wsnTree=CicsTest,
    ou=RAH,
    o=IBM Hursley,
    c=UK
ibm-wsnName: CICSDB2instance
javaClassName: com.ibm.db2.jcc.DB2SimpleDataSource
ibm-wsnEntryType: SerializableLeaf
objectclass: ibm-wsnEntry
entryOwner: access-id:cn=admin
entryOwner: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK
aclentry: group:cn=anybody:normal:rsc
aclentry: access-id:cn=CICSUser,ou=RAH,o=IBM Hursley,c=UK :
          object:ad:normal:rwsc
aclentry: access-id:cn=CICSSystems,ou=RAH,o=IBM Hursley,c=UK :
          object:ad:normal:rwsc
```

If you were going to define multiple JDBC entries, the first half will not need to be done again (as the `jdbc` node will already have been defined). You merely need to change the (blue) initial `ibm-wsnName` settings and execute.

Observe the red `javaClassName:` setting of `com.ibm.db2.jcc.DB2SimpleDataSource`. It is this which 'turns' the DataSource object into the required DB2SimpleDataSource instance.

This file is executed in the usual fashion for the addition of a LDAP leaf:

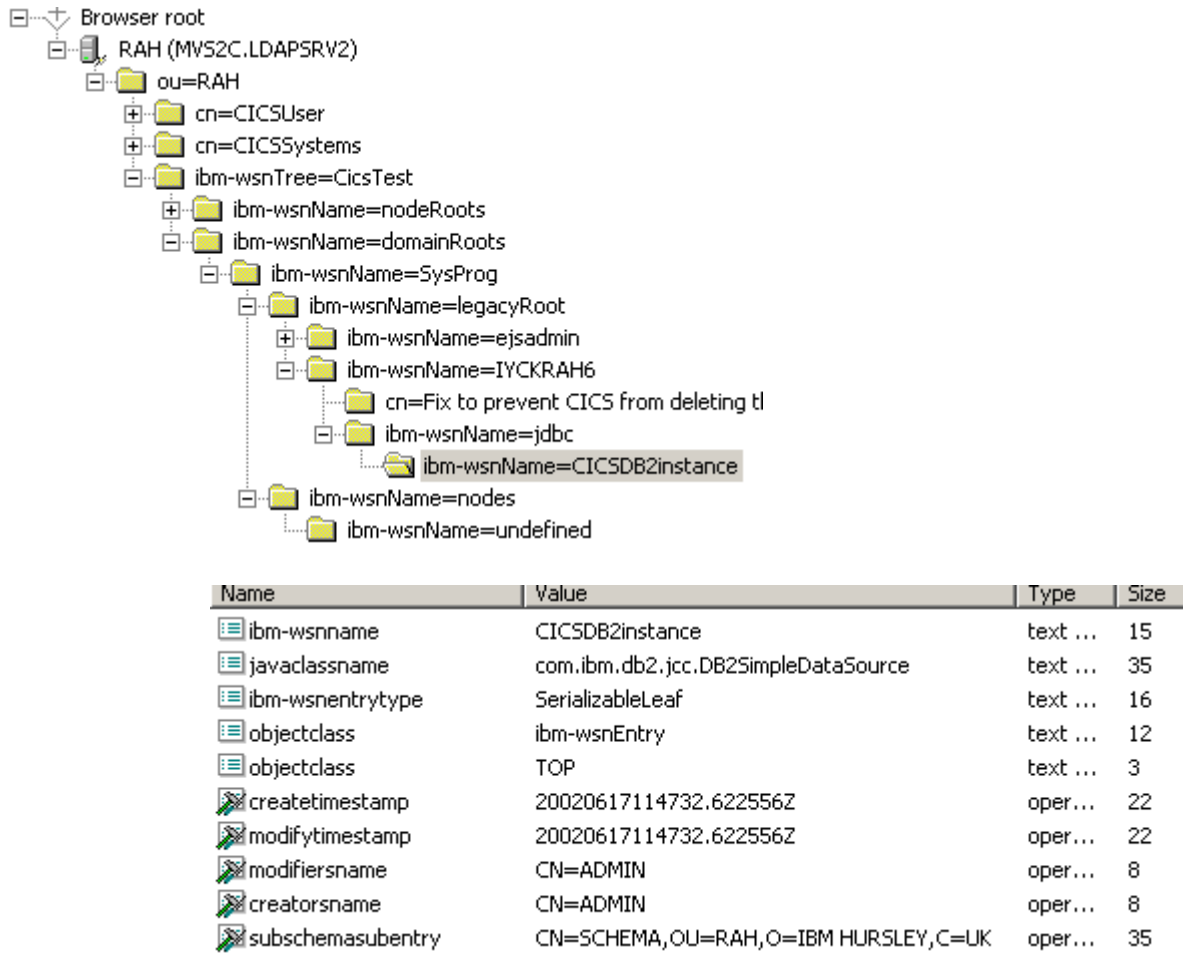
```
ldapadd     -v                                    \
            -h winmvs2c.hursley.ibm.com -p 2389   \
            -D "cn=admin"               -w secret \
            -f Myjdbc.ldif
```

LDAP leaf creation and JNDI verbs

I have done all the LDAP node defintions for the JDBC entry through an utility definition so that the correct ACLs (permissions) are set. This means that the subsequent Bind operation for LDAP is going to use the **`rebind`** verb (rather than `bind`) as this is the flavour of JNDI operation that requires the definition to exist.

Results of the node creation

After the LDIF script has been run, the Directory tool shows the created entry. Observe that it has not been 'filled in' with any data suitable for recreating the CICS `DB2SimpleDataSource` object:



Name	Value	Type	Size
ibm-wsnname	CICSDB2instance	text ...	15
javaclassname	com.ibm.db2.jcc.DB2SimpleDataSource	text ...	35
ibm-wsnentrytype	SerializableLeaf	text ...	16
objectclass	ibm-wsnEntry	text ...	12
objectclass	TOP	text ...	3
createtimestamp	20020617114732.622556Z	oper...	22
modifytimestamp	20020617114732.622556Z	oper...	22
modifiersname	CN=ADMIN	oper...	8
creatorsname	CN=ADMIN	oper...	8
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	oper...	35

Figure 23: LDAP JDBC node structure

However, the required `DB2SimpleDataSource` class has been recorded.

Publishing the Object to LDAP

This definition has merely set the environment to contain the information required to initialise the `DB2SimpleDataSource` object. The population of this information has to be done from within the owning CICS region.

The act of population is to save a java stringified version of a `DB2SimpleDataSource` object. This information is used in populating the `DataSource` object and it should look something like:

```
ACED0005 73720016 6A617661 782E6E61    *    ¼f ?sr ?javax.na
6D696E67 2E526566 6572656E 6365E8C6    *    ming.ReferenceF¦
9EA2A8E9 8D090200 044C0005 61646472    *    Pó¿Tì?? ?L ?addr
73740012 4C6A6176 612F7574 696C2F56    *    st ?Ljava/util/V
6563746F 723B4C00 0C636C61 73734661    *    ector;L ?classFa
63746F72 79740012 4C6A6176 612F6C61    *    ctoryt ?Ljava/la
6E672F53 7472696E 673B4C00 14636C61    *    ng/String;L ¶cla
73734661 63746F72 794C6F63 6174696F    *    ssFactoryLocatio
6E71007E 00024C00 09636C61 73734E61    *    nq ~ ?L ?classNa
6D657100 7E000278 70737200 106A6176    *    meq ~ ?xpsr ?jav
612E7574 696C2E56 6563746F 72D9977D    *    a.util.Vector+ù}
5B803BAF 01020003 49001163 61706163    *    [Ç;»?? ?I ?capac
69747949 6E637265 6D656E74 49000C65    *    ityIncrementI ?e
6C656D65 6E74436F 756E745B 000B656C    *    lementCount[ ?el
656D656E 74446174 61740013 5B4C6A61    *    ementDatat ?[Lja
76612F6C 616E672F 4F626A65 63743B78    *    va/lang/Object;x
70000000 00000000 02757200 135B4C6A    *    p       ?ur ?[Lj
6176612E 6C616E67 2E4F626A 6563743B    *    ava.lang.Object;
90CE589F 1073296C 02000078 70000000    *    É+Xƒ?s)l?  xp
0A737200 1A6A6176 61782E6E 616D696E    *    ?sr ?javax.namin
672E5374 72696E67 52656641 64647284    *    g.StringRefAddrä
4BF43CE1 11DCC902 00014C00 08636F6E    *    K(<ß?_+? ?L ?con
74656E74 7371007E 00027872 00146A61    *    tentsq ~ ?xr ¶ja
7661782E 6E616D69 6E672E52 65664164    *    vax.naming.RefAd
6472EBA0 079A0238 AF4A0200 014C0008    *    drdá•Ü?8»J? ?L ?
61646472 54797065 71007E00 02787074    *    addrTypeq ~ ?xpt
000B6465 73637269 7074696F 6E740042    *     ?descriptiont B
44423220 44617461 536F7572 63652077    *    DB2 DataSource w
69746820 64656661 756C7420 55524C20    *    ith default URL
666F7220 75736520 62792043 49435320    *    for use by CICS
5472616E 73616374 696F6E20 53657276    *    Transaction Serv
65727371 007E0009 74000C6C 6F67696E    *    ersq ~ ?t ?login
54696D65 6F757474 00013070 70707070    *    Timeoutt ?0ppppp
70707074 0024636F 6D2E6962 6D2E6462    *    pppt $com.ibm.db
322E6A63 632E4442 32446174 61536F75    *    2.jcc.DB2DataSou
72636546 6163746F 72797074 0023636F    *    rceFactorypt #co
6D2E6962 6D2E6462 322E6A63 632E4442    *    m.ibm.db2.jcc.DB
3253696D 706C6544 61746153 6F757263    *    2SimpleDataSourc
65                                     *    e
```

The code required to run within the owning CICS region to populate the JNDI entry will be something like:

```
Context ctx = new InitialContext() ;

// Get the JDBC leaf name

String contextDataBaseName = "com.ibm.cics.datasource.name" ;
String dataSourceName = System.getProperty(contextDataBaseName);

// Create the DB2SimpleDataSource Object to be published

DB2SimpleDataSource ds = new DB2SimpleDataSource() ;

// Annote the object (if required) or set some more defaults

ds.setDescription(
 "DB2 datasource with default URL for use by CICS Transaction Server"
                 );

// Publish the Object to JNDI when the definitions exist

ctx.rebind(dataSourceName,ds.getReference()) ;
```

The JNDI **`rebind`** verb is used to update the JNDI entry in the LDAP server.

If the **`bind`** verb was used instead of the `rebind`, then the LDAP utility definition would not create the final leaf node (the bit after the last \ which is the `CICSDB2instance` part in my example) as the `bind` operation will do this.

The published `DB2SimpleDataSource` object does not contain an addressing URL for the database. This defaults to a Default URL of `jdbc:default:connection` which is the recommended setting for access to DB2 via JDBC from within CICS.

This code fragment was created using a Java Development Tool - I called it `Publish_to_JNDI_via_LDAP_rebind` - and compiled within that environment.

The resulting `Publish_to_JNDI_via_LDAP_rebuild.class` was FTPed to my MVS region, and a RDO (case dependant) entry of

```
 PROGram         : JNDIPUBR
 Group           : RAHJAVA
 DEscription     : PUBLISH TO JNDI/rebind
 Language        :                    CObol | Assembler | Le370 |
 RELoad          : No                 No | Yes
 RESident        : No                 No | Yes
 USAge           : Normal             Normal | Transient
 USElpacopy      : No                 No | Yes
 Status          : Enabled            Enabled | Disabled
 RSl             : 00                 0-24 | Public
 CEdf            : Yes                Yes | No
 DAtalocation    : Below              Below | Any
 EXECKey         : User               User | Cics
 COncurrency     : Threadsafe         Quasirent | Threadsafe
REMOTE ATTRIBUTES
 DYnamic         : No                 No | Yes
 REMOTESystem    :
 REMOTEName      :
 Transid         :
 EXECUtionset    : Fullapi            Fullapi | Dplsubset
JVM ATTRIBUTES
 JVM             : Yes                No | Yes
 JVMClass        : Publish_to_JNDI_via_LDAP_rebind
                 :
                 :
                 :
                 :
 JVMProfile      : DFHJVMPR
JAVA PROGRAM OBJECT ATTRIBUTES
 Hotpool         : No                 No | Yes
```

was used via `CECI LINK PROG(JNDIPUBR)` to update the JNDI entry.

All sorts of strange failures can occur if the correct properties and settings are not correct.

Within the DFHJVM member of DFHJVMPR, I had the following items defined (amongst others):

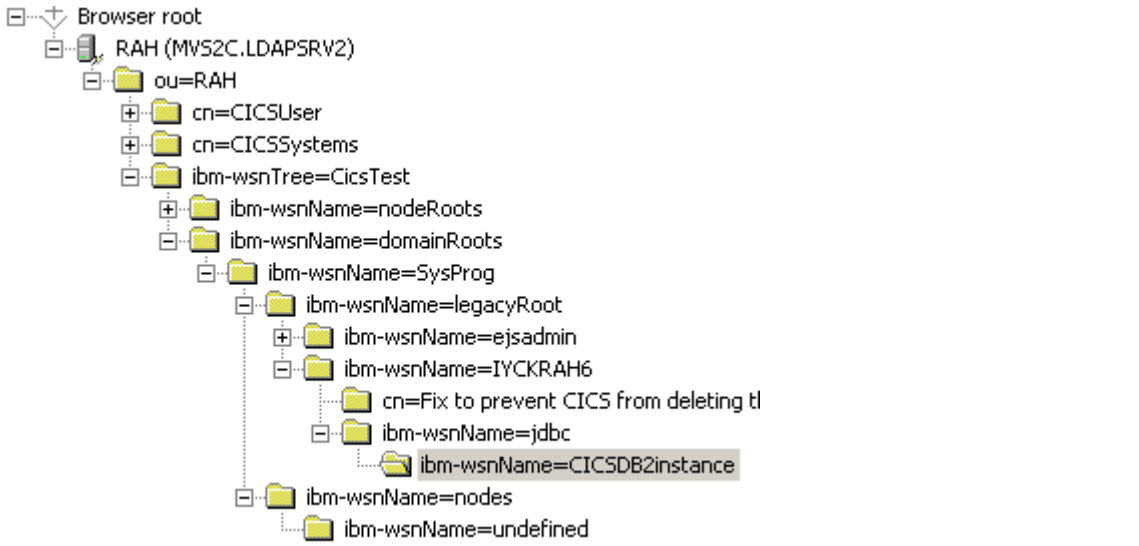
```
LIBPATH            :/usr/lpp/java131s/J1.3\
                   :/usr/lpp/java131s/J1.3/bin\
                   :/usr/lpp/java131s/J1.3/bin/classic\
                   :/usr/lpp/db2710/db2710/lib\

TMSUFFIX           /usr/lpp/db2710/db2710/classes/db2sqljruntime.zip:\
                   /usr/lpp/db2710/db2710/classes/db2j2classes.zip:\

CLASSPATH          :/usr/lpp/java131s/J1.3\
                   :/usr/lpp/java131s/J1.3/bin\
                   :/usr/lpp/java131s/J1.3/bin/classic\
                   :/usr/lpp/db2710/db2710/lib\
                   :/usr/lpp/db2710/db2710/classes\
```

and within the relevant `system.properties` file:

```
com.ibm.cics.datasource.name=IYCKRAH6/jdbc/CICSDB2instance
```

Results of Publication

After the rebind code has been run within CICS, the Directory Browser will contain the stringified data for the creation of the CICS `DB2SimpleDataSource` object which will be used for the creation of the `DataSource` object:



Name	Value	Type	Size
ibm-wsnname	CICSDB2instance	text ...	15
ibm-wsnentrytype	SerializableLeaf	text ...	16
objectclass	ibm-wsnEntry	text ...	12
objectclass	javaSerializedObject	text ...	20
objectclass	JAVAOBJECT	text ...	10
objectclass	TOP	text ...	3
javaserializeddata	AC ED 00 05 73 72 00 16 6A 61 76 61 ...	binar...	609
javaclassname	com.ibm.db2.jcc.DB2SimpleDataSource	text ...	35
createtimestamp	20020617114536.156983Z	oper...	22
modifytimestamp	20020617114536.156983Z	oper...	22
modifiersname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C...	oper...	37
creatorsname	CN=CICSUSER,OU=RAH,O=IBM HURSLEY,C...	oper...	37
subschemasubentry	CN=SCHEMA,OU=RAH,O=IBM HURSLEY,C=UK	oper...	35

Figure 24: Results of Rebind operation

>>>>>>>>>>>>>>>END of document<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<